RAG using ReAct framework from scratch

Adrien Giget, Théo Ripoll, Nicolas Fidel, Quentin Fisch

1. Project description (~400 words)

The goal of this project is to create a tool using the Retrieval Augmented Generation (RAG) pattern as well as the ReAct framework, from scratch. In this case, from scratch means we do not want to use libraries such as Langchain, since it would just take 20 lines of code to have very satisfying results.

Our tool will be able to receive a document containing text (a PDF file) and answer questions using information present in the file. This is the RAG part. The ReAct framework goal is to use both “reasoning” and “acting” to try and mimic some of the human behavior when trying to induce, search, and update a thought. The framework will allow the LLM behind the RAG to choose between multiple actions and create a more human-like chain of thought. You can find examples and sample outputs on this page: https://www.promptingguide.ai/techniques/react. In order to provide the context of the file when trying to find an answer, we will use a vector database that will hold the embeddings of the content of the file. This database will then be consulted by comparing the embeddings of both the input (what the LLM thinks is relevant to search for) and the content of the file. Then, if anything is found during this search, it will be given to the prompt when talking again with the LLM.

Why is this project interesting? Nowadays, it seems like everybody calls themselves IA masters without even knowing what is the difference between a sigmoid and a ReLu. We want to use this project to understand more about NLP, RAGs and the different methods that will come into play (such as prompting, vector search, which LLM to use…). Surely there exists better solutions than the one we will create (as mentioned above: Langchain), but we believe it is strictly useless to simply use a library and have no idea of what is going on behind it. Indeed, when we tested Langchain to see the results, we were impressed by the efficiency, but had a hard time trying to understand what was REALLY happening (not trying to convince ourselves and change subject five minutes later).

Such a tool could be very powerful in any company where there is a need to find information quickly in a dense and verbose database of files. However, it may be often interesting for companies who are using chatbots for this type of tasks to know what is going on, where is the information retrieved from and how can they be certain the reasoning behind the answer is correct. The ReAct framework would be a good assistant for this particular need.

2. Project background (~400 words)

The concept of Retrieval Augmented Generation (RAG) represents a significant leap in NLP and artificial intelligence. By integrating retrieval into the generative process, RAG models have enhanced the capabilities of language models to provide more accurate, context-rich, and informed responses. This approach has been pivotal in fields like chatbots, search engines, and information retrieval systems.

The inception of RAG (https://arxiv.org/pdf/2311.04177.pdf) can be traced back to the advancements in transformer-based models, notably with the success of models like BERT and GPT. The integration of external knowledge bases into the generation process

marks the core of RAG's methodology. This has enabled models to not only generate based on pre-trained knowledge but also to pull in relevant information from external sources in real-time.

Concurrently, the ReAct framework (https://arxiv.org/abs/2210.03629 and https://react-lm.github.io/) has emerged as a promising tool in enhancing the decision-making capabilities of AI models. It combines reasoning and acting, allowing models to emulate more closely human thought processes in decision-making. The framework's application in AI spans various domains, from automating complex tasks to improving the interactivity of chatbots. Its significance lies in its ability to dynamically choose actions based on the context, thus leading to more natural and human-like interactions.

In the industrial context, companies like OpenAI, Google (https://cloud.google.com/blog/products/databases/introducing-sample-genai-databases-retrieval-app?hl=en), and others have been at the forefront of applying RAG and ReAct-like frameworks. These technologies have been instrumental in developing sophisticated AI assistants, enhancing search engine capabilities, and automating complex decision-making processes in business environments.

Despite these advancements, there remains a gap in the understanding and application of these technologies at a granular level. Many implementations rely on pre-packaged libraries, offering limited insight into the underlying mechanics. This gap highlights the importance of projects like ours, which aim to build these systems from scratch for deeper comprehension and customized application.

Our project is situated within this evolving landscape, aiming to not only utilize these cutting-edge technologies but also to contribute to the understanding of their inner workings. By building a RAG model using the ReAct framework from scratch, we intend to delve into the intricacies of NLP, AI decision-making processes, and the practical challenges of integrating complex frameworks into usable tools. This endeavor is particularly relevant in the current AI-driven era, where understanding the nuances of such technologies is as crucial as their application.

3. Project steps (Bullet points)

The following tries to recap what has to be done for the project to be successful, but it does not provide any time or importance order.

- ☑ Use libraries (Langchain) to see potential results (Quentin)
- ☐ “Play” with different LLMs and try to see which seems to be best to fit to the task (Adrien, Quentin)
    - Open source ? GPT-4 through OpenAI’s API ? …
    - What is the best prompting method ? What can we do to know which action to take ?
    - …
- ☑ Implement a PDF loader to get a document and extract informations (Théo)

    - Load file
    - Cut text in chunks
    - Compute embeddings
- ☐ Choose and setup vector database (Théo, Nicolas)
    - ChromaDB ? Pinecone ? MongoDB Atlas Vector Search ? …
- ☐ Create the main tool with the connection between the LLM and the database using functions from the ReAct framework (Everyone)
- ☐ Build a front to have a more valuable product using Gradio or Streamlit (not defined yet as this is not the most important part)

4. First results (min. 200 words)

The first results are pretty encouraging, even if a lot still needs to be done. Firstly, even if it might not seem like it, a lot has been done in the background to be sure to understand how the ReAct framework is working, how to improve our prompt engineering skills, etc. It takes time and might not be as obvious as other results which are more straightforward to produce.

Regarding Langchain, we, of course, don't want to use it in our final product, but we gave it a try to know what it is possible to do and what we should aim for. We have conducted tests on the wikipedia database (instead of using pdf files), and the results are great. The notebook in the repository is self explanatory.

The PDF loader is working well and was not a big challenge. The code is also in the repository.

We "played" with several LLMs (GPT-4, Mistral, Zephyr, Llama), and we think it would be better practice to use open source tools for this project, even if the openai Python library seems to be the easiest way to use in this project. Indeed, the library has a function call feature, which can return a function to use (in a list of given functions) using the context of the prompt. This is very powerful and can help us a lot, especially regarding prompt engineering and being sure the model will constantly give functions from the ReAct framework. As of today, we still want to try to use an open source LLM, preferably available HuggingFace. Our current best option is Mistral, but we need to do deeper research on this topic.

The vector database will not be an issue at all. They are easy to set up and the use of the associated library will be straightforward. We still have not made any decision on which one we will use since this is not the current priority.

Github repo: https://github.com/QuentinFISCH/wikipedia-react-rag