Chatbot RAG, chat with your documents

Cynthia García González

ITIC Barcelona

[cynthia@garcia.barcelona](mailto:cynthia@garcia.barcelona)

# Abstract

We present a chatbot architecture built upon the Retrieval Augmented Generation (RAG) framework, capable of responding to user queries about specific knowledge outside the training data origins, extending the powerful capabilities of Large Language Models (LLMs). This architecture is developed using text fragments from Hugging Face documentation as the data source and utilizes Hugging Face for extracting pre-trained models and Langchain for connecting these models. Our approach achieves coherent results without retraining the selected pre-trained models, demonstrating the potential of RAG-based chatbots in knowledge retrieval and generation tasks.

1. Introduction

The advent of Large Language Models (LLMs) has revolutionized the field of natural language processing, enabling unprecedented capabilities in text generation, language translation, and question answering. However, these models are often limited by their training data, struggling to generalize to out-of-distribution inputs or respond to queries that require external knowledge. Recently, the Retrieval Augmented Generation (RAG) [1] framework has emerged as a promising approach to address these limitations, by combining the strengths of LLMs with the ability to retrieve and incorporate external knowledge.

In the context of RAG, LLMs are used as generators, producing text based on a given prompt or query. However, these generators are augmented with a retrieval mechanism, which searches a large database of text fragments to provide relevant information that can inform the generation process. This allows the model to respond to queries that

require knowledge beyond its training data, and to generate more accurate and informative text.

Despite the potential of RAG, its application is still in its early stages, and several challenges remain to be addressed. For instance, the retrieval mechanism must be able to efficiently search a massive database of text fragments, and the generator must be able to effectively incorporate the retrieved information into its output. Furthermore, the model must be able to generalize to new domains and topics, and to adapt to changing user preferences and requirements.

1. Dataset

Given the constraints posed by limited computational resources, utilizing the m-ric/huggingface\_doc [3] dataset, which comprises around 1.5 million text fragments extracted from the Hugging Face documentation, was not feasible. The sheer volume of data in this dataset demands significant processing power and time, particularly during the vectorization process required to transform the textual data into a format suitable for machine learning models. However, due to the limited availability of computational resources, it wasn't practical to allocate the necessary processing power and time for this task.

Considering these constraints, a more manageable alternative was sought. A fragment from "The Hobbit" [2] was chosen as an alternative dataset. This decision was motivated by the smaller size of the text fragment compared to the extensive Hugging Face documentation dataset. By working with a smaller dataset, the processing requirements are significantly reduced, making it more feasible to conduct experiments and analyses within the available resource constraints.

It's important to acknowledge that while using the fragment from "The Hobbit" helps mitigate the resource limitations, it also comes with trade-offs. The dataset may not encompass the same breadth and depth of information as the Hugging Face documentation dataset. Consequently, the model trained on this smaller dataset may not be as robust or accurate in handling queries outside the scope of "The Hobbit" fragment. However, it serves as a practical compromise given the resource constraints, allowing for experimentation and exploration within the available computational limits. Additionally, it's worth noting that individuals with access to more substantial computational resources could potentially leverage the comprehensive m-ric/huggingface\_doc dataset to conduct more specific and detailed analyses.

1. Architecture

The architecture of the Advanced Retrieval Augmented Answer Generation (RAG) solution [Figure 1], developed by Hugging Face and Langchain, comprises several essential components collaborating to deliver accurate and pertinent responses to user inquiries. This architecture delineates into two primary segments: chatbot learning and user query processing.

**Chatbot Learning**

Within this initial phase, the chatbot accesses the dataset using a Langchain file loader.

**Document Splitting into Chunks and Tokenization**

In this stage, documents undergo segmentation into tokenized fragments, termed chunks, which are utilized by the reading model for generating the final response. This process employs the Langchain recursive method, which incorporates various separators to ensure consistency for the reading model.

**Retrieval**

The tokenized and segmented data are stored in a FAISS vector database, which initially converts them into vector representations or embeddings using the *thentlper/gte-small* model [4]. The architecture of the *thentlper/gte-small* model comprises 6 transform layers, featuring a hidden state and input vector size of 768, 12 attention heads, and a feedforward size of 3072.

This database employs the same embedding model to embed the user's query and, via a similarity search, retrieves the documents closest to it among all those stored in the vector database. Cosine similarity [6] is employed to compute the distance between vectors, determining the similarity between two vectors as the cosine of their relative angles. This method enables comparison of vector directions irrespective of their magnitude, necessitating the normalization of all vectors to rescale them into uni-norm.

**Reranking**

Once the initial retrieval phase retrieves a set of candidate documents based on the user's query, a reranking step is employed to further refine the selection. Reranking involves subjecting the retrieved documents to a more sophisticated evaluation, enhancing the precision and relevance of the results.

In our framework reranking begins by utilizing a more advanced retrieval model, namely Colbertv2, to reassess the relevance of the candidate documents. Colbertv2 enables a deeper analysis of the interaction between query tokens and document tokens.

Each document retrieved during the initial retrieval stage undergoes reembedding using Colbertv2. This process involves computing more nuanced interactions between the query tokens and the tokens of each document. Subsequently, the reembedded documents are scored based on their relevance to the user's query. Colbertv2 assigns a score to each document, indicating its likelihood of being relevant. These scores are then used to rank the documents in order of relevance.

Finally, only the top\_k documents with the highest relevance scores are retained, where k is a predefined threshold determined based on factors such as the user's preferences or the application's requirements.

This integration of Colbertv2 into the reranking process aims to enhance the accuracy and effectiveness of document selection, ensuring that the results better align with the user's information needs.

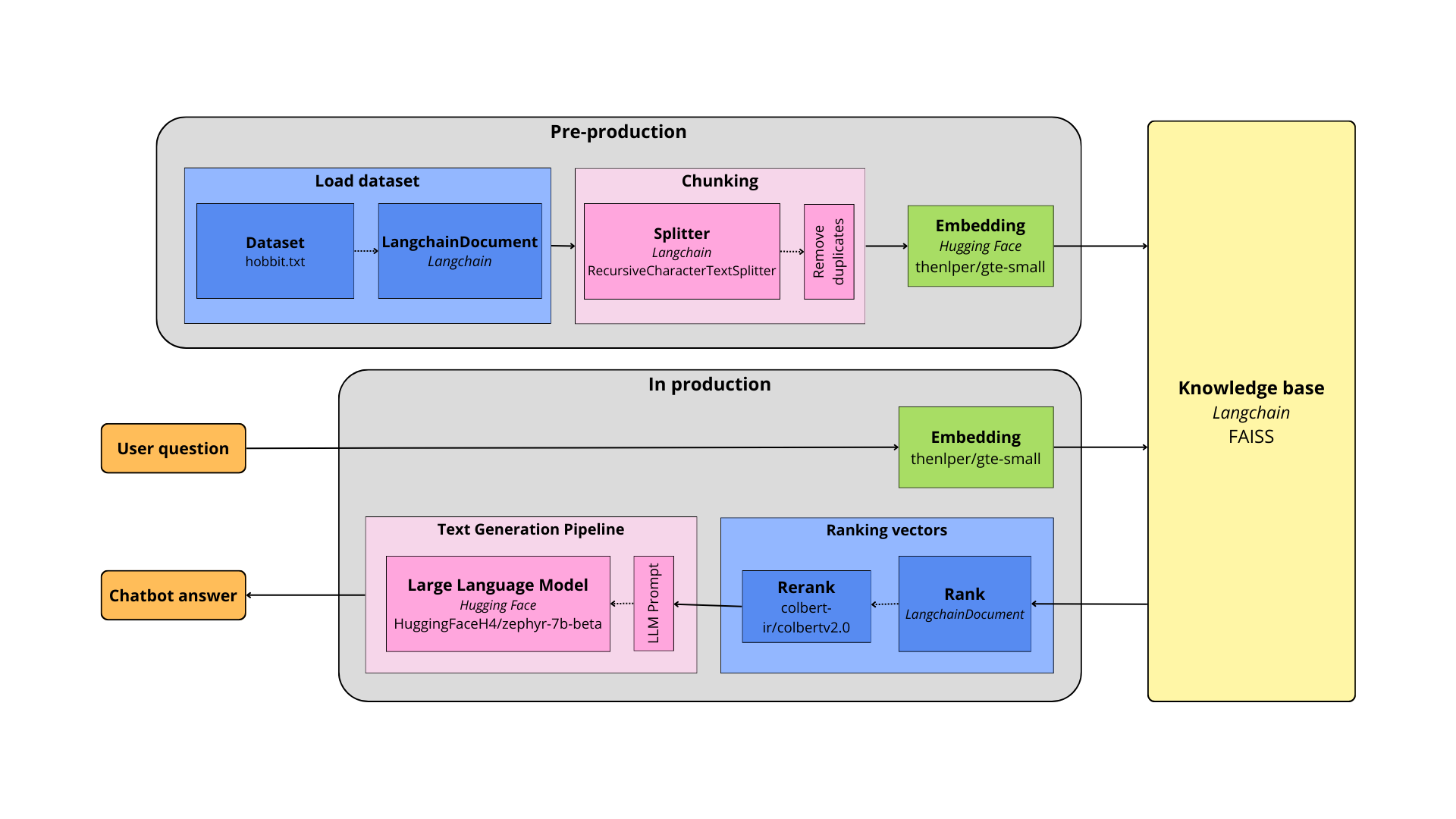
Initially, the architecture utilized in the retrieval phase did not incorporate Colbertv2. However, integrating Colbertv2 into the reranking process has proven instrumental in enhancing the accuracy and effectiveness of document selection, ultimately leading to improved results.

Figure 1. An illustration of our Chatbot architecture.

**Reading Model**

This component selects the relevant retrieved fragments and generates a coherent, pertinent response to the user's query. The reading model is defined using the pre-trained language model "HuggingFaceH4/zephyr-7b-beta" [5] accessible from Hugging Face Transformers. This model has undergone pre-training on extensive text corpora and can produce consistent, relevant answers to user queries.

In essence, Hugging Face's advanced RAG solution architecture [Figure 1] encompasses four pivotal components: the knowledge base, the retrieval component, chunking, and the reading model. The knowledge base constitutes the documents utilized by the system as a source of information. The retrieval component leverages vector representations of the documents to identify the most relevant fragments. Chunking partitions the documents into smaller fragments, while the reading model generates a coherent, relevant response to the user's query. This architecture is constructed using the Hugging Face Datasets, Langchain, and Transformers libraries, configured with specific parameters tailored to each component.

1. Results

**Changes in dataset**

Given the constraints of limited resources, the primary bottleneck affecting system performance is

the processing time required when dealing with large datasets. Consequently, the m-ric/huggingface\_doc dataset was not used. Instead, a text file containing a fragment of the first chapter of The Hobbit was utilized. This decision was made to reduce the processing time by working with a smaller dataset.

**Retrieval results**

In the results section, it is imperative to highlight a significant aspect of the experimentation: the interrogation of the chat system sans the use of the language model (LLM). Instead, the system exclusively relied on an embedding model for processing inquiries, employing a document retrieval mechanism to furnish relevant documents in response. This approach was chosen to evaluate the system's proficiency in handling queries solely through semantic embeddings and document matching, bypassing the complex language generation capabilities of the LLM.

Given the limited scope of the initial retriever, comprising only three documents, the system returned two related documents for each inquiry. This constraint was instrumental in simulating a scenario where a relatively small corpus was available for retrieval, akin to real-world applications where comprehensive datasets might be unavailable or computationally prohibitive.

The interrogation process encompassed a diverse set of inquiries, spanning both English and Spanish languages [Figure 2]. This deliberate choice aimed to assess the system's adaptability and effectiveness across linguistic variations. The questions posed were meticulously crafted to elicit nuanced responses, probing various aspects of the provided text fragment.

"*What is a hobbit*?": This inquiry seeks to ascertain the system's ability to define and contextualize the concept of a "hobbit," a central theme within the text fragment.

"¿*Cómo se describe la casa del protagonista*?" (How is the protagonist's house described?): This query delves into descriptive elements within the text, evaluating the system's aptitude in extracting specific details regarding the protagonist's residence.

Gráfico, Gráfico de barras

Descripción generada automáticamente"*What is said about the reputation of the Baggins family*?": By posing this question, the aim is to gauge the system's capability in discerning and summarizing information pertaining to the reputation of the Baggins family as conveyed in the text.

Figure 2.

"¿*Qué características físicas y culturales distinguen a los hobbits según el texto*?" (What physical and cultural characteristics distinguish hobbits according to the text?): This inquiry aims to test the system's capacity to extract and synthesize information regarding the physical and cultural attributes that distinguish hobbits, drawing from the textual context provided.

Each question serves as a litmus test for the system's proficiency in semantic understanding, document retrieval, and response generation, shedding light on its capabilities in comprehending and addressing inquiries in varied linguistic contexts, thereby underlining its potential utility in multilingual applications.

**Reranking results**

Despite limited resources and slower-than-expected response times due to constraints, the implementation of reranking has proven to be an effective strategy for improving the quality of responses generated by the system.

The inclusion of Colbertv2 in the reranking process has been particularly beneficial as it enables more precise selection of relevant documents for information extraction. This is because Colbertv2, encode both the query and documents into high-dimensional vectors that capture semantic relationships between words and phrases. In contrast, search similarity in a vector database is based on the comparison of low dimensionality vectors representing documents and queries. This comparison may be less accurate, since low-dimensional vectors may not fully capture semantic relationships between words and phrases.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteBy incorporating a reranking process, we have been able to better identify and prioritize the most relevant documents for information extraction [Figure 3]. This has led to a significant improvement in retrieval performance and, consequently, higher quality responses for our users and has allowed us to overcome the limitations imposed by our resources and response times, delivering more accurate and efficient information retrieval for our users.

Figure 3. Result of reranking

1. Discussion

In summary, the arrival of Large Language Models (LLMs) has revolutionized the field of natural language processing, but they still face limitations when it comes to generalizing out-of-distribution inputs and responding to queries that require external knowledge. The Retrieval Augmented Generation (RAG) framework has emerged as a promising solution to address these limitations, combining the strengths of LLMs with the ability to retrieve and incorporate external knowledge.

The RAG-based chatbot architecture presented in this document demonstrates an effective solution for improving the quality of responses generated by the system, even with limited resources and slower-than-expected response times. The inclusion of Colbertv2 in the reranking process has been particularly beneficial, allowing for more precise selection of relevant documents for information extraction. This has led to a significant improvement in retrieval performance and, consequently, higher quality responses. Additionally, the ability to generate coherent and accurate responses without retraining the selected pre-trained models demonstrates the potential of RAG-based chatbots in knowledge retrieval and generation tasks.

Despite the potential of RAG, challenges remain to be addressed, such as the need for the retrieval mechanism to efficiently search a massive database of text fragments and for the generator to effectively incorporate the retrieved information into its output. Furthermore, the model must be able to generalize to new domains and topics and adapt to changing user preferences and requirements. Nevertheless, the RAG-based chatbot architecture presented in this document represents an important step towards overcoming these limitations and developing more effective and accurate Chatbots.

1. References

[1] RAG Langchain Hugging Face: <https://huggingface.co/learn/cookbook/advanced_rag>

[2] Dataset: <https://github.com/jblazzy/LOTR/blob/master/hobbit.txt>

[3] Cloud dataset: <https://huggingface.co/datasets/m-ric/huggingface_doc>

[4] Embedding model: <https://huggingface.co/thenlper/gte-small>

[5] LLM model: <https://huggingface.co/HuggingFaceH4/zephyr-7b-beta>

[6] Cosine similarity: <https://pytorch.org/docs/stable/generated/torch.nn.CosineSimilarity.html>