Docu-Bot: AI assisted user support

Jiří Chudoba[1,*], *Michal* Chudoba[2,**], and *Lukáš* Hejtmánek[3,***]

[1]FZU, Institute of Physics of the Czech Academy of Sciences, Prague, Czech Republic
[2]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
[3]Institute of Computer Science, Masaryk University, Brno, Czech Republic

Abstract. Users may have difficulties to find the needed information in the documentation for products, when many pages of documentation are available on multiple web pages or in email forums. We have developed and tested an AI based tool, which can help users to find answers to their questions. The Docubot uses Retrieval Augmentation Generation solution to generate answers to various questions. It uses github or open gitlab repositories with documentation as a source of information. Zip files with documentation in a plain text or markdown format can also be used for input. Sentence transformer model and Large Language Model generate answers. Different LLM models can be used. For performance reasons, in most tests we use the model Mistral-7B-Instruct-v0.2, which fits into the memory of the Nvidia T4 GPU. We have also tested a larger model Mixtral-8x7B-Instruct-v0.1, which requires more GPU memory, available for example on Nvidia A100, A40 or H100 GPU cards. Another possibility is to use the API of OpenAI models like gpt-3.5-turbo, but the user has to provide his/her own API access key to cover expenses.

1 Introduction

Modern computing infrastructures often provide various APIs and configuration options that can be confusing for users. When multiple documentation sources are spread across web pages or email forums, users may struggle to locate the information they need. Additionally, projects with different versions often implement features inconsistently, creating further complexity. For instance, the dCache project keeps support tickets private, leading to repeated inquiries that could otherwise be resolved through shared knowledge.

Our solution, Docu-Bot, is an AI-driven tool designed to address these issues by quickly delivering accurate answers drawn from relevant documentation. By implementing Retrieval Augmented Generation (RAG), Docu-Bot provides targeted responses, minimizing support workload and improving user access to information.

2 Retrieval Augmented Generation

Our solution leverages the Retrieval Augmented Generation (RAG) [1] approach, enabling the LLM to process information relevant to a user's question and present it in a more user-

*e-mail: jiri.chudoba@cern.ch
**e-mail: michal.chudoba.praha@gmail.com
***e-mail: xhejtman@ics.muni.cz

friendly format. This method allows for accurate answers without requiring extensive fine-tuning for each specific project. Additionally, the retrieval function identifies the source documents used to construct each answer, guiding users to the original content if they wish to explore further.

2.1 Used Configuration

We selected the **Mistral-7B-Instruct-v0.2** model as the default LLM, paired with the **sentence-transformers/all-MiniLM-L6-v2** model for document embedding. This configuration fits within the memory constraints of widely available GPUs (such as our T4 with 16GB), but the implementation permits any models conforming to OpenAI API documentation[1] to be used. We have therefore also tested OpenAI models, including **GPT-4-turbo** as the LLM and **text-embeddings-3-small** and found them to be marginally better. The code regarding our setup is available on GitHub[2].

2.2 Pipeline

The RAG operates as a two-step pipeline. Firstly, requested seed documentation must be embedded for future usage by the pipeline. This is done by dividing individual documents into fixed-sized chunks (we have division by Markdown or RST headers with an upper limit of 500 characters), embedding them, and storing the vectors into the database with a document ID. Documents are stored under a document ID separately for quick access. Secondly, when retrieving documents to support the user's question, the same embedding process is used, and based on cosine similarity, the most promising chunks are retrieved. The document IDs associated with these chunks are then used to retrieve the final documents for the LLM to use. This is showcased in Figure 1

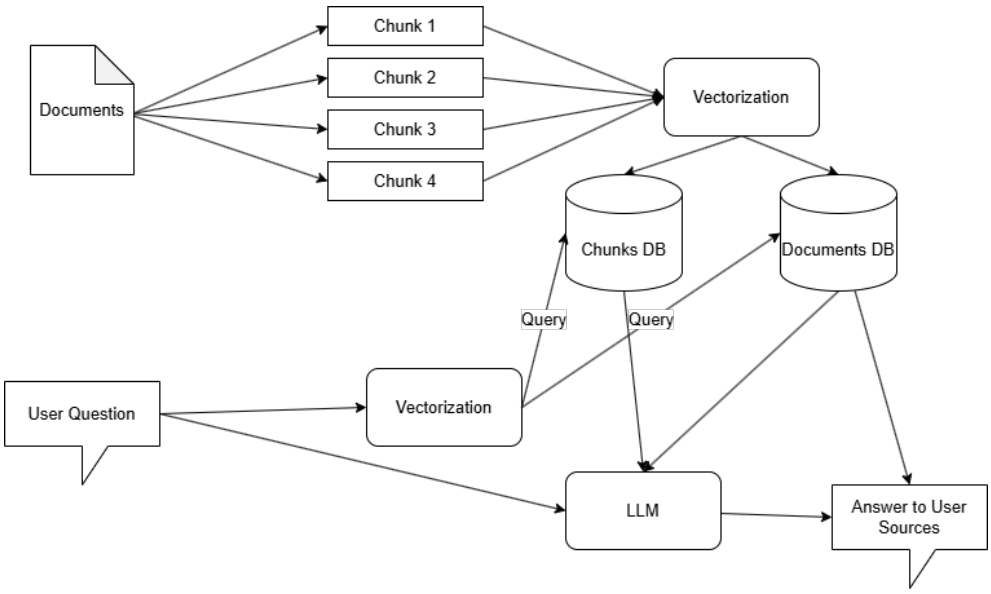

Figure 1. Schema of our RAG pipeline

[1] https://platform.openai.com/docs/api-reference/introduction
[2] https://github.com/jinymusim/Docu-Bot

3 Used Documents

For document sources, we selected a combination of public and private documentation as well as support tickets. To manage different project versions, the tool pulls from public GitHub and GitLab repositories, where each branch represents a unique version. For better answer precision, each version is stored in a separate database. The end user can specify which repositories and branches the pipeline should retrieve to generate the final answer, tailoring responses to the most relevant information for their needs.

4 Examples

To demonstrate the Docu-Bot's functionality, we used the dCache project GitHub repository, specifically testing it with the latest release (version 9.2). The applications interface is built using the *Gradio* Python package, which serves as a user-friendly front-end.

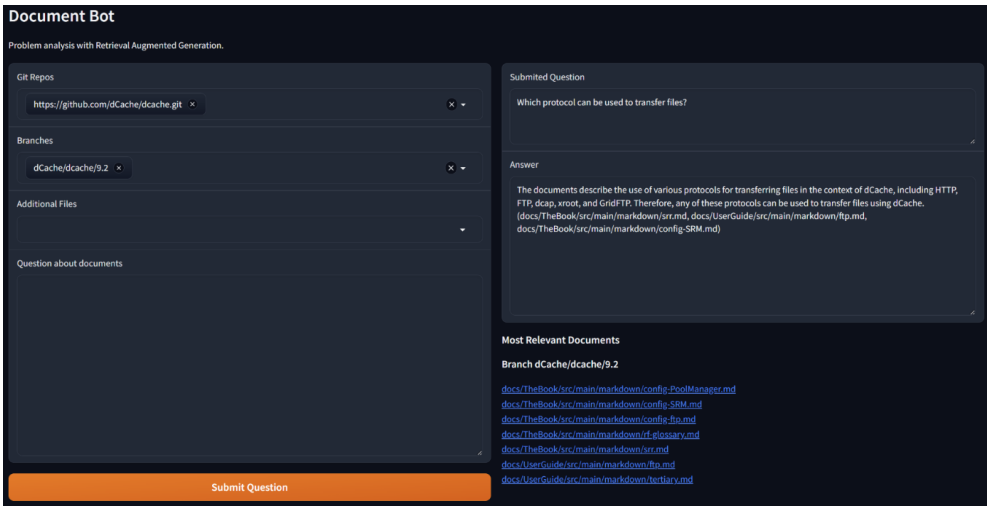

Figure 2. Example of the interface with answered question.

In Figure 2, the tools main capabilities are highlighted. Users can choose Git repositories, specify branches within the repositories, and include additional sources (such as ZIP files containing email conversations and support tickets). The Docu-Bot then utilizes the RAG pipeline to process the query, extracting relevant information and providing an answer with links to the most important sources.

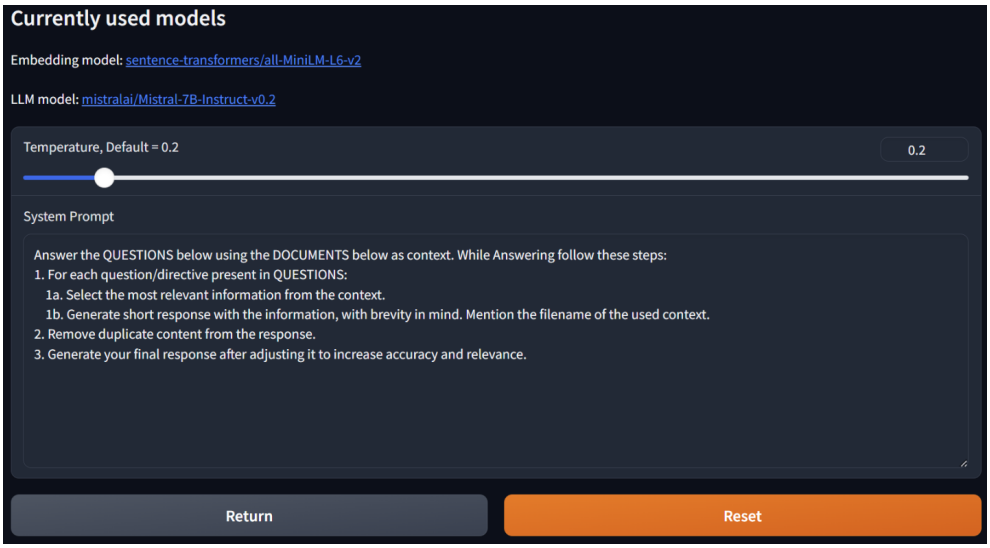

Figure 3. Configurable prompt and model temperature

Figure 3 further illustrates customization options available to users. Users can adjust prompts and model temperature settings, which impact the models inference process. Research shows [2] that prompt tuning significantly enhances model performance, helping users optimize output based on their specific needs.

5 Summary

This paper demonstrates how integrating a large language model (LLM) with document sources using the Retrieval Augmented Generation (RAG) framework can significantly enhance user support systems. By embedding relevant documentation and support tickets, the tool allows for accurate, citation-based responses to user inquiries. Through examples, we illustrate how user questions are addressed with references to the original documents, ensuring transparency and reliability in the answers provided.

6 Acknowledgments

Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

[1] Lewis et al., Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)
[2] Lester et al., The Power of Scale for Parameter-Efficient Prompt Tuning, 2021