University *of Ljubljana*
Faculty *of Computer and Information Science*

Conversational Agent with Retrieval-Augmented Generation

Katarina Velkov, Nejc Krajšek, Luka Sabotič

Abstract

abstract

Keywords
Conversational agent, Retrieval-Augmented Generation

Advisors: Aleš Žagar

Introduction

Large language models (LLM) rely solely on their pre-trained knowledge to generate responses. These models have proved to be powerful but they are prone to generating hallucinated or inaccurate information. Furthermore their use in specialized fields has proved to be challenging as LLMs are trained mostly on publicly available data and are not updated after new data arises. That is why the need for more advanced models has emerged. Retrieval-Augmented Generation (RAG) enables LLMs to retrieve more relevant information from various external databases and web sources during conversations. This process allows for more accurate, domain-specific and up-to-date responses.

The aim of this project is to build a conversational agent using the RAG technique. We wil use an existing large language model, expand it to be able to query linked domain-specific databases. We will evaluate the performance of our model by comparing it to the original LLM to discover improvements our Implementation brought and also other RAGs available online to inspect the overall quality of our work.

Related work

The concept of Retrieval-Augmented Generation (RAG) was first introduced in 2020 by [**?**], who proposed a framework that combines the strengths of large language models (LLMs) with external knowledge retrieval. Since then, RAG has seen significant advancements, particularly in the context of domain-specific applications and conversational agents.

In the paper "Towards Optimizing a Retrieval Augmented Generation using Large Language Model on Academic Data" [**?**], the authors explore various optimization techniques for RAG in the academic domain. They introduce several enhancements, such as Multi-Query, Child-Parent Retriever, Ensemble Retriever, and In-Context Learning, to improve the performance of RAG systems. Their experiments demonstrate that incorporating multi-query techniques significantly boosts retrieval effectiveness, especially in domain-specific contexts like university study programs. The authors also propose a novel evaluation approach, the RAG Confusion Matrix, to assess the effectiveness of different RAG configurations.

Another relevant work is the survey titled "Retrieval-Augmented Generation for Large Language Models: A Survey," [**?**], which provides a comprehensive overview of RAG's evolution. The survey categorizes RAG into three paradigms: Naive RAG, Advanced RAG, and Modular RAG. It highlights the importance of retrieval, generation, and augmentation techniques in enhancing the performance of RAG systems. The survey also discusses the challenges and future directions of RAG, such as improving robustness, handling long contexts, and integrating multimodal data.

Methods

For our project, we aim to develop a conversational agent that leverages RAG to provide accurate responses by dynamically retrieving information from the provided databases. We are going to focus on the programming and debbugging domain, trying to create a model that will correctly produce, complete and corrrect the given code. We will provide it with public Github [**?**] and Stack overflow [**?**] data that our model will query and find similar examples which it will use to help the user.

The proposed methodology involves the following steps:

Implementation plan

Our plan is to create a pipeline structured as an advanced RAG. It looks as following:

- **Indexing** - preprocessing of database data into a uniform format (plain text). Text is then segmented into chunks which are smaller and easier to work with than with original documents. We will use an embedding model to store these chunks into a vector database for later use.
- **Pre-retrieval** - optimizing the user query to improve its indexing and retrieved chunks
- **Retrieval** - embedding the query into the vector database and retrieving similar vectors
- **Post-retrieval** - process retrieved chunks to create a quality context to be passed to LLM. Emphasize most relevant context and guard against overloading the LLM with information.
- **Generation** - combine the (improved) user query with the retrieved context and pass it to the LLM to get the desired answer

As our base LLM, we will use ChatGPT 3 and OpenAI API to integrate it into our model. We will also need a way to embed external data and store it into a vector database into which we will embed user queries. To do that, we are going to use either Chroma or Pinecone. We might need to use other resources for query optimization as well, but we are not yet sure what will be needed.

Results

Discussion

Acknowledgments