swisscom

Neural Wave Hackathon Problem Statement

Version	Date	Person	Adjustments / comments
0.1	25.09.2024	S. Taillefert, D. Menini	Initial draft
0.2	30.09.2024	S. Taillefert, D. Menini	Revised draft for submission
1.0	14.10.2024	S. Taillefert, D. Menini	Final version

Author: Stefano Taillefert, Davide Menini Created on: 25.09.2024 C2 General

Content

1 Introduction to the Domain / Motivation for the Problem 3

2 Problem Statement 3

3 Explanation of the Data 4

4 Potential Solution and Validation 5

5 Additional Resources 6

5.1 Contacts during the Hackathon 6

1 Introduction to the Domain / Motivation for the Problem

Swisscom is the largest IT provider in Switzerland, employing more than 5'000 people in its IT, Network & Infrastructure (INI) department out of the more than 16'000 employees. Swisscom operates six data centres in Switzerland, with an IT infrastructure which comprises over 80,000 virtual machines and around 6,000 servers[1].

A sizeable part of our business involves interaction with customers, which can be enhanced and streamlined with the usage of chatbots and similar AI agents (such as Sam, the digital assistant used in the My Swisscom application). Our end goal is to provide the best customer support possible, which is why we are investing into the latest AI technologies to further delight our customers and improve our processes.

2 Problem Statement

As large language models (LLMs) are increasingly applied in various scenarios, it's becoming clear that they are not all-powerful. When used to generate dialogues, these models often produce hallucinations, leading to inaccurate responses. While LLMs store a vast amount of knowledge within their neural network parameters, this information is fixed at the time of training, making retraining impractical. However, by properly enhancing the model, it can still be a powerful tool—this is where Retrieval-Augmented Generation (RAG) comes in.

RAG is an AI framework that improves the quality of responses generated by LLMs by integrating external knowledge sources. By incorporating these external references, the model can augment its text generation with up-to-date and reliable information, increasing both the accuracy and relevance of its output.

At Swisscom, we are empowering the next generation of chatbots through RAG. We use curated knowledge bases consisting of public data from the Swisscom website and documents from other internal databases to provide external knowledge to our chatbot solutions, such as the new generation Sam, offering customers first hand support on a wide range of tasks.

While Sam uses RAG, it is not a simple RAG bot but a more powerful agentic solution also able to perform actions to actively help customers, and not only provide information. However, the latter is still a valid use case: many employees must search through internal documents, and being able to chat with their own data is a simple yet efficient way of retrieving information using natural language. This type of question-answering chatbots is what this challenge is about.

Your task is to build a simple question answering chatbot using RAG over the public Swisscom website data that we provide.

The first step will be data indexing, as to perform retrieval, you need to build a search engine. Data indexing consists of 3 steps:

1. Split documents into smaller **chunks** of given size. Chunks can also overlap with each other to better retain information

[1] https://www.swisscom.ch/en/about/governance/organisation-and-structure.html

2. Create **embeddings** of the document chunks using an embedding model
3. Store document chunks and their embeddings into the **vector store**, a database specialized on storing and retrieving data through semantic search

The size of document chunks plays a crucial role in determining both the quantity of search results that can fit within the LLM context window and the quality of the embeddings.

Breaking documents into smaller chunks offers several advantages:

- **Increased Retrieval:** smaller document chunks enable the retrieval of more documents within the LLM context window, enhancing the comprehensiveness of search results
- **Enhanced Semantic Embeddings:** smaller chunks often result in better-quality semantic embeddings. This can lead to more precise and relevant search outcomes

However, there's some trade-offs to consider:

- **Potential Information Gaps:** when documents are divided into smaller chunks, there's a risk that some essential information may be fragmented across multiple pieces
- **Lost in the Middle Issue:** if too many documents are retrieved, the LLM will be overloaded with information and may decrease attention towards documents in the middle

Once the vector store has been populated with the data, you are ready to query it at runtime whenever your chatbot receives a question from the user. In particular:

1. **Retrieval**: use the user question to query the vector store and retrieve the top *K* document chunks
2. **Augmentation**: inject the top *K* document chunks as context for the LLM prompt
3. **Generation**: use the LLM to generate an answer to the user question

3 Explanation of the Data

The provided dataset contains:

- Raw HTML webpages from the Swisscom website in multiple languages (German, French, Italian, and English)
- Cleaned-up webpages in Markdown format, ready to be indexed in the vector store

Each file is a JSON containing the following fields:

- Content (str): the raw/processed data
- Source (str): the source of the content (an URL most of the times)
- Title (str | None): the title of the webpage (optional)
- Language (str | None): the language of the webpage (optional)

The dataset can be downloaded at the following URL: https://swisscom-my.sharepoint.com/:u:/p/stefano_taillefert/EegWIyF8835PuUXsyuzmGGsBcxu7gFVcJVhyOpLVhZ_g4A?e=nsivZN

The password is: Hackathon2024

swisscom

4 Potential Solution and Validation

Our reference RAG pipeline uses the following technologies and ideas:

- We use OpenSearch as vector store
- We use Cohere as embeddings API. The max chunk size that it's allowed by the provider is 2024 characters, thus our chunks are limited to this size. The overlap between subsequent chunks is roughly 10% of the chunk size
- We use Antrophic's Claude 3.5 Sonnet as LLM with a temperature of 0.3
- At runtime we retrieve the top 10 chunks using similarity search through OpenSearch's FAISS search engine
- We use a LLM chain composed of (1) a summarization step to summarize the chat history into a standalone question to be used as retrieval query, and (2) a generation step to produce the final answer from the retrieved context and the original user question
- Our framework of choice for LLM chains is LangChain

Your chatbots will be validated and rated against our own solution on a small evaluation dataset. We will use a judge LLM to decide whether your chatbot's answer or our reference is better, according to the following criteria:

- Language consistency: does the chatbot answer in the language spoken by the user? And if the user asks to speak in another language, how does it behave?
- Answer consistency: does the chatbot correctly answer a question that relates to a previous exchange in the conversation? And if the new question has nothing to do with it, how does it behave?
- Relevance: is the answer inherent in the question?
- Factuality: is the answer based on the retrieved context?
- Correlation: is the reference mentioned in the answer?

Note: human judges will oversee the validation process. The list is subject to changes and interpretation.

5 Additional Resources

RAG frameworks:

- LangChain documentation: https://python.langchain.com/docs/introduction/
- Build your RAG pipeline: https://python.langchain.com/docs/tutorials/rag/
- LlamaIndex documentation: https://docs.llamaindex.ai/en/stable/

Improvement ideas:

- Finetune an embedding model: https://huggingface.co/blog/train-sentence-transformers#:~:text=Sentence%20Transformers%20is%20a%20Python,%2C%20paraphrase%20mining%2C%20and%20more.
- Add contextual information to chunks: https://www.anthropic.com/news/contextual-retrieval
- Use similarity search together with other search techniques (e.g. BM25) and merge the results with Reciprocal Rerank Fusion (RRF)
- Use a reranker model to sort the results: https://www.pinecone.io/learn/series/rag/rerankers
- Work on prompts and use more complicated chains or agents (see what LangChain has to offer for advanced RAG)
- Combine all the previous ideas, and come up with your customized pipeline

5.1 Contacts during the Hackathon

Stefano will be on site for the kick off on Friday and the closing ceremony on Sunday as a Swisscom representative and for general support.

Together with the participants, we will organize a way for them to contact us for specific questions about the project (group chat on Teams, Discord, or similar).

For any questions or further clarifications, feel free to contact us by email, Teams, or phone (if urgent).

Stefano Taillefert, stefano.taillefert@swisscom.com, +41 79 738 97 09 (general support)

Davide Menini, davide.menini@swisscom.com, +41 78 977 99 90 (domain expert)