CMPE-259 Natural Language Processing

Project Report

1 Introduction

Natural Language Processing (NLP) has experienced a significant boom with the introduction of Large Language Models (LLMs), and developments in this field have rapidly increased. With each passing day, these models are becoming more robust and efficient at interpreting user queries and providing appropriate and relevant answers. Inspired by this, I plan to develop a Large Language Model-based Virtual Assistant (VA). This VA is designed to answer questions related to the Service Agreement and Customer Agreement of Amazon Web Services (AWS). The business use case for this VA is to assist both customers and employers using AWS services in understanding its policies more clearly. Both agreements are available on the AWS website, with links to the Customer Agreement and Service Agreement. These pages can be saved as PDF files and used to build a Retrieval-Augmented Generation (RAG) pipeline to train the LLM to answer specific questions related to these agreements. With the rise of large language models, many different models have been developed by various companies. Some of these models are open source, while others require payment to use their services. For this project, I used Ollama-llama, Ollama-mistral, gemma-2b, and OpenAI-GPT-3.5-Turbo. Ollama is a platform that manages and runs open-source LLMs locally. OpenAI developed GPT, which is a paid model and is one of the best LLMs available on the market. To create a RAG pipeline, the PDFs will be stored in a vector database to provide context to the LLM while generating responses based on user queries. Top-K similar chunks of data will be provided to the LLM based on the user query. However, the LLM response cannot always be trusted as there is a very high chance the model can hallucinate and generate inaccurate answers. To rectify this, I included a grader to validate the LLM response. Results include the responses generated by using different LLM models and embedding models for the given user questions. In addition, all the responses are logged in a CSV file, and a UI is created using Streamlit to make things much cleaner.

2 Project Architecture

Project architecture can be broken down into two parts: 1. Storing embeddings in Vector database, 2. Fetching context, Generating response, and Evaluation

- **Storing Embeddings in Vector database**: PDF's are converted into text chunks and embeddings for the text chunks are generated using different embedding models. I pulled open-source models from ollama. In addition, I used OpenAI embedding and Nomic embedding models too. Embeddings are stored in **SKLearnVectorStore** vector database. Scikit-learn is an open-source collection of machine learning algorithms, including some implementations of the k nearest neighbors. SKLearnVectorStore wraps this implementation and adds the possibility to persist the vector store in json, bson (binary json) or Apache Parquet format.
- **Fetching context, Generating response, and Evaluation**: Based on user query, top-k relevant content is fetched from vector database to provide context to the llm and it is included in the llm prompt template. This is will help llm to generate an accurate response. LLM will generate a response with the provided context, but if the content fetched from the vector store is not good, then there is a chance of model to hallucinate and generate an inaccurate answer. To check this, I included a grader in the pipeline. Grader will check the response with the context fetched from vector store and checks if the facts are matching. Fact checking will let us know if the model is generating a sophisticated or hallucinated response.

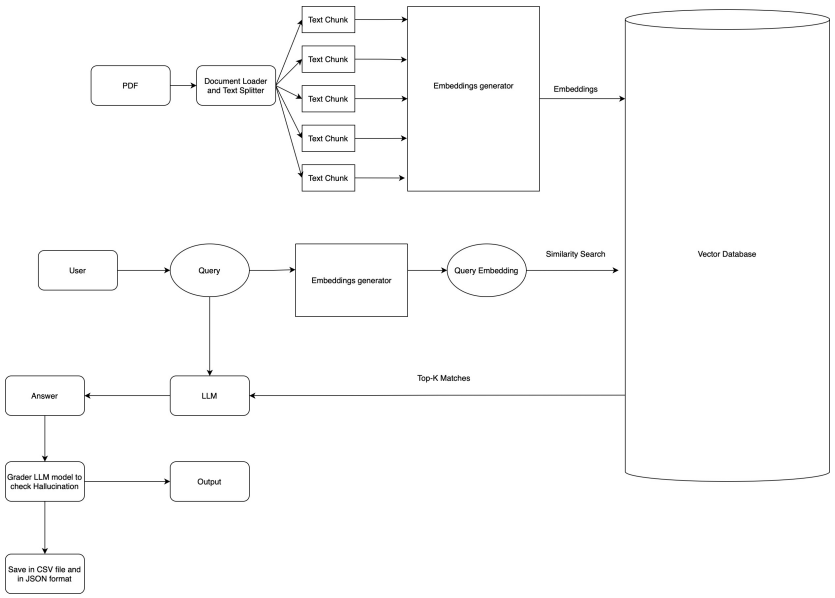

Figure 1: Project architecture

3 Tech Stack

- LangChain, Ollama, OpenAI Models, Streamlit, SKLearnVectorStore

4 Embedding

4.1 Ollama Embedding

Ollama manages a lot of open-source embedding models. Firstly, we have to install ollama locally and use `ollama pull <embedding-model-name>`.

- **paraphrase-multilingual**: This is a sentence transformer model that can be used for semantic search and clustering tasks. This is a 278M parameter model and uses BERT transformer architecture. Similar to BERT, it has 768 dimensional dense vector space.
- **mxbai-embed-large**: This is a SOTA model developed by mixedbread.ai. This is 334M parameter model with architecture similar to BERT. It outperforms commercial models like text-embedding-3 of OpenAI and matches the performance of model 20X its size.
- **all-minilm**: This is a small model with 22.6M parameters and architecture similar to BERT.

4.2 Nomic Embedding

Nomic Embeddings are provided by Nomic Atlas. Nomic provides embeddings for different types of data forms like Image, Text, and Dimensionality reduction. Text embeddings of nomic are so sophisticated and is on top of the Massive Text Embedding Benchmark(MTEB) table with score of 62.39%. I used **nomic-embed-text-v1** model which is specialized for retrieval, similarity, clustering, and classification tasks. This has a context length of 8192 and dimension of 768.

4.3 OpenAI Embedding

OpenAI embedding model is called as text-embedding-3. There are two subversions of this model and they are called as small and large. This is also a SOTA model and very sophisticated in generating embeddings. Performance of this model is as close to nomic embedding with MTEB score of 62.3% and context length of 8191. In contrast to the above mentioned models, this is a paid model and a dollar is charged for 62,500 pages.

5 Large Language Models

For this project, I used **gpt-3.5-turbo** and **ollama-llama3** models. Ollam-llama is a open-source model and gpt-3.5-turbo is a commercial model provided by OpenAI.

- Ollama-llama: I used llama3.2:3b-instruct-fp16 model, provided by meta and this is a 3 billion parameter model. This is a sophisticated model that can outperform gemma2 and phi 3.5-mini. This is good at summarization, following instructions, and promt rewriting. For this project, we need a concise and accurate text based on the fetched context from vector store.
- gpt-3.5-turbo: This is a commercial model provided by OpenAI. We need to get some credits and generate an API key to use this model. This is a relatively good model that can be used to generate text based on the context provided and this is a very robust model with very less chance of hallucinating.

6 Methodology

- The PDFs are converted to text chunks using LangChain RecursiveCharacterTextSplitter. On the UI, user will have an option to select the embedding model, llm model, a text area to type in his question, and a button to generate answer.
- Once the inputs are given and clicked on button, the code pipeline will be triggered. Firstly, similarity seach will be performed, and top-k matches related to the user query are fetched from vector store. I used 3 to be my k value. Now this context is added to the rag prompt and formatted to get complete prompt. Based on the prompt and the chosen llm, a response will be generated.
- Now the generated response will be graded using gpt-3.5-turbo model to check if the model is hallucinating. A grader promt is given, where the model will check the generated response with facts fetched from the vector store. After the matching, a grading score between 0 to 1 is shown on the screen along with the explanation for this score. Finally, the generated response and response evaluation is stored in a csv file along with the chosen models.

7 Results

- Emedding model: nomic-embed-text-v1, Model: gpt-3.5-turbo
  - Question: what is AWS re:Post Private
  - Response: AWS re:Post Private is a service provided by Amazon Web Services (AWS) that allows users to store and process content in AWS regions outside of the region where the service is being used. This means that users can access and utilize AWS re:Post Private to manage their content in different regions for various purposes. It is important to note that the use of AWS re:Post Private is subject to the Terms of Use specific to this service, which can be found in the private re:Post platform. Additionally, users are responsible for all content that is processed by AWS re:Post Private and should be aware that their content may be stored in regions other than where they initially accessed the service.
  - Evaluation: Grading Model: gpt-3.5-turbo. Score: 1. Explanation: The student's answer is perfectly grounded in the facts provided. The answer accurately describes AWS re:Post Private as a service that allows users to store and process content in AWS regions outside of the region where the service is being used. The mention of Terms of Use specific to the service and the responsibility of users for the content processed align with the facts.
- Emedding model: mxbai-embed-large, Model: gpt-3.5-turbo
  - Question: what is aws mailing address of Australia

– Response: The AWS mailing address for Australia is as follows: Amazon Web Services Australia Pty Ltd (ABN: 63 605 345 891), Level 37, 2-26 Park Street, Sydney, NSW, 2000, Australia

– Evaluation: Grading Model: gpt-3.5-turbo. Score: 1. Explanation: The STUDENT ANSWER contains information about AWS Builder ID that is not supported by the provided FACTS. There is no mention of AWS Builder ID in the given text, and the answer relies on general knowledge rather than being grounded in the provided information.