Help-Mate AI: Search Systems _ C61_Sundaralingam

1. Introduction

This project demonstrates creating a search system named “HelpMate AI” with the help of insurance policy document using RAG techniques.

2. Problem Statement

The goal of the project is to build a robust generative search system capable of effectively and accurately answering questions from a policy document.

We will be using a single long life insurance policy document for this project.

3. Methodology

The project should implement all the three layers effectively. It will be key to try out various strategies and experiments in various layers to build an effective search system. Let’s explore what we need to do in each of the layers.

1. **The Embedding Layer:** The PDF document needs to be effectively processed, cleaned, and chunked for the embeddings. Here, the choice of the chunking strategy will have a large impact on the final quality of the retrieved results. So, we need to make sure that we try out various strategies and compare their performances. Another important aspect in the embedding layer is the choice of the embed- ding model. we can choose to embed the chunks using the OpenAI embed- ding model or any model from the Sentence Transformers library on HuggingFace.

2. **The Search Layer:** Here, we need to design at least 3 queries against which you will test your system. You need to understand and skim through the document, and accordingly come up with some queries, the answers to which can be found in the policy document. Next, we need to embed the queries and search your ChromaDB vector database against each of these queries. Implementing a cache mechanism is also mandatory. Finally, we need to implement the re-ranking block, and for this we can choose from a range of cross-encoding models on HuggingFace.

3. **The Generation Layer:** In the generation layer, the final prompt that we design is the major component. We need to make sure that the prompt is exhaustive in its instructions, and the relevant information is correctly passed

to the prompt. We may also choose to provide some few-shot examples in an attempt to improve the LLM output.

4. System Layers

- **Reading & Processing PDF File:** We will be using pdfplumber to read and process the PDF files. pdfplumber allows for better parsing of the PDF file as it can read various elements of the PDF apart from the plain text, such as, tables, images, etc. It also offers wide functionalities and visual debugging features to help with advanced preprocessing as well.

- **Document Chunking:** The document contains several pages and contains huge text, before generating the embeddings, we need to generate the chunks. Let's start with a basic chunking technique, and chunking the text with fixed size.

- **Generating Embeddings:** Generates embedding with SentenceTransformer with all-MiniLM-L6-v2 model.

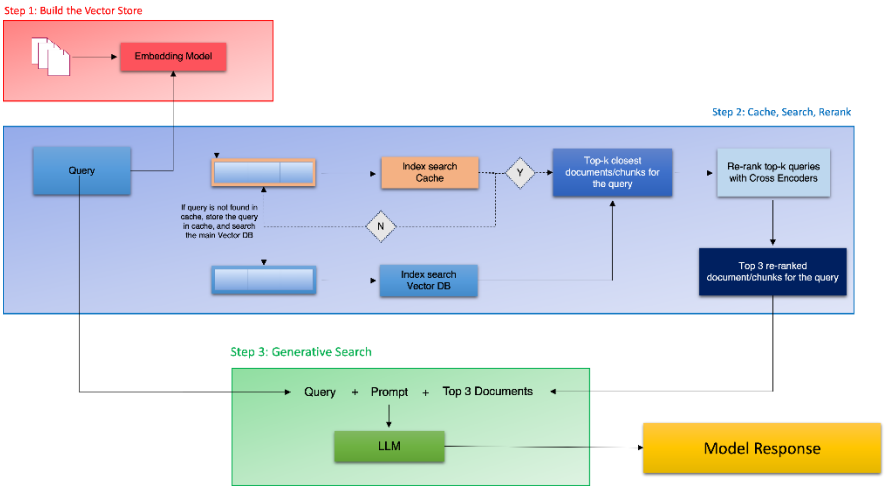
- **Store Embeddings In ChromaDB:** In this section we will store embedding in ChromaDB.

- **Semantic Search with Cache:** In this section we will introduce cache collection layer for embeddings.

- **Re-Ranking with a Cross Encoder:** Re-ranking the results obtained from the semantic search will sometime significantly improve the relevance of the retrieved results. This is often done by passing the query paired with each of the retrieved responses into a cross-encoder to score the relevance of the response w.r.t. the query.

- **Retrieval Augmented Generation:** Now we have the final top search results, we can pass it to an GPT 3.5 along with the user query and a well-engineered prompt, to generate a direct answer to the query along with citations.

5. System Architecture



6. Queries Screenshot:

```
query = 'what are the limitations in this policy ?'
df = search(query)
df = apply_cross_encoder(query, df)
df = get_topn(3, df)
response = generate_response(query, df)
print("\n".join(response))
```

```
The limitations in the policy documents are as follows:

| Limitations |
|-------------------------------------|
| Not valid for employee use          |
| Only officers of the group          |

Citations:
1. Policy Name: Household Policy
   Page Number: Page 13
2. Policy Name: Group Policy
   Page Number: Page 16
```

```
[63] query = 'Can we do cashless transaction for this policy ?'
df = search(query)
df = apply_cross_encoder(query, df)
df = get_topn(3, df)
response = generate_response(query, df)
print("\n".join(response))
```

```
Yes, this policy does support cashless transactions. You can proceed with cashless transactions for this policy.

---

**Citations:**
- Policy Name: Household Policy
- Page Number: Page 13
```

```
[64] query = 'Who can be nominee for this policy ?'
df = search(query)
df = apply_cross_encoder(query, df)
df = get_topn(3, df)
response = generate_response(query, df)
print("\n".join(response))
```

```
The nominee for the policy can be any household member, employee, or person other than an officer of the agency to which this Group Policy is subject.

**Response:**
The nominee for this policy can be any household member, employee, or person other than an officer of the agency to which this Group Policy is subject.

**Citations:**
- Policy Name: Group Policy
- Page Number: Page 13, Page 16
```

```
[ ] Start coding or generate with AI.
```