A dive into Human Rights and Law with an RAG and Question Answering System

Dinara Kurmangaliyeva, Emel Safarini, Zeynep Sila Kaya, Ruslan Nuriev, Sezgi Cobanbas

Project Introduction and Objectives

This project focuses on developing a Retrieval-Augmented Generation (RAG) and Question Answering system capable of parsing and interpreting historical documents such as the Magna Carta. The system aims to assist students and educators by providing accurate, context-aware responses to human rights and legal principles queries by leveraging modern natural language processing methodologies. The application of the RAG model can be extended beyond Human Rights and Law, but more practically in the area of medicine, where an RAG system trained on medical journals and research papers could help doctors and patients navigate complex medical information.

Objectives

In this project we are aiming to achieve the following objectives:

1. **Semantic Splitting of Legal Texts**:

   Experiment with multiple text chunking methods to find the most efficient way to apply semantic splitting to legal texts. This includes:

   - **Recursive Character Text Splitter (Naive Chunking)**: Dividing text based on a fixed character count.
   - **Interquartile-Based Splitting**: Utilizing the interquartile range to determine text chunks.
   - **Percentile-Based Splitting**: Splitting text where the difference between sentences exceeds a certain percentile.
   - **Manual Chunking:** Chunking the document manually based on some programmer-defined conditions
2. **Embedding Model Integration**:
   - Select and implement an appropriate embedding model to process the semantically split text for further analysis.
3. **Construction of a RAG Model**:
   - Develop a Retrieval-Augmented Generation model by integrating a pre-trained embedding model. This involves processing the text and storing it in a vector database, enabling efficient retrieval for answering queries.
4. **Creation of a Synthetic Dataset for testing**:
   - Create a synthetic dataset to evaluate a Large Language Model (LLM) to enhance its ability to understand and respond to questions related to human rights law.

By achieving these objectives, the project will contribute a valuable tool for the study and interpretation of legal texts, enhancing accessibility and comprehension of significant historical and legal documents.

Model Implementation

In this section, we outline the detailed steps and methodologies employed in the development of our Retrieval-Augmented Generation (RAG) and Question Answering system.

1. Environment Setup

First, we need to install the necessary libraries for text parsing, RAG model construction, and integration of embedding models.

2. Document Parsing

Next, we parse the document using Tika to extract the text content from the PDF. Parsing the document is critical for extracting the raw text content from legal documents. By this step, we want to be sure that our text format can be processed by our NLP model.

After we split a large text document into smaller, manageable chunks using the `Recursive` Character Text Splitter, interquartile chunker, and percentile chunker. from the LangChain library. This is an essential step in preparing the text for further processing in an RAG system. We defined each chunk will have a maximum of 256 characters and there will be no overlap between consecutive chunks.
Using regular expressions, which allows us to search, match, and manipulate strings based on patterns, we split the text content of the parsed PDF document into chunks whenever it encounters a pattern that matches the beginning of the article. By this technique, each chunk now corresponds to a specific article, making retrieving particular sections easier.

3. Model template

Firstly, we will define the template for our model. By template, we mean:
• If the model can't find a related context to our question in the text, it will output "I don't know"
• Otherwise, it will take the related context and augment it with the user-defined question

4. Database Creation

RAG models retrieve the context from vector databases. In this part, we will define 4 vector databases for each of our chunks. To transfer our text file into numbers in the vector databases, we will use the "text-embedding-3-large" embedding model from OpenAI.

5. Defining the model

Our model uses a template from the previous step: a chat prompt that includes instructions for the AI model, a placeholder for the user's query, and a placeholder for the context. The

ChatPromptTemplate object is then created from this template, which can be used to generate chat prompts. RAG models retrieve the context from vector databases.

FAISS is a library for efficient similarity search and clustering, and OpenAIEmbeddings is a library for generating embeddings of text using OpenAI's models. We set up a vector store and retrieved text chunks based on their semantic similarity, using OpenAI's embeddings and FAISS. After that, create an instance of the ChatOpenAI class, which is wrapped around OpenAI's chat models. To sum up, ChatOpenAI is a class provided by the langchain_openai library, which allows us to interact with OpenAI's chat models programmatically, by creating an instance of ChatOpenAI with no arguments.

These are the main bodies of our models. We first defined the context by setting it to the relevant retriever from our vector base and in the same way, we defined the question by setting it to the user-defined question. Then these two will go through our pre-defined prompt. As we know from earlier, the prompt will retrieve the context and augment it with the question before feeding it to our base model. When the base model receives these two models, it will generate an answer. The response is converted into a string using the StrOutputParser.The answer can solely be related to our database.

6. **Model Testing with Random Questions**

For comparing the different chunking methods and in general, for comparing the models, we created some random questions based on our text and got answers from these models. We also created some questions that are not in the document to see if it outputs "I don't know".

For example, if we ask " What is the natural and fundamental group in society? " to the model we built from the naive chunking method we are gonna get " The natural and fundamental group in society is the family." as an answer. Also from the percentile chunking method, we can get "The natural and fundamental group in society is the family, according to the context provided." as an answer.

Model Evaluation and Results

In the first part, we defined some metrics for evaluating an RAG model, or a measurement to compare the generated answer with the ground truth answer. We'll see those 4 metrics in detail:

- **Context Precision** - it is a metric that evaluates whether all of the ground-truth relevant items present in the contexts are ranked higher or not. Ideally, all the relevant chunks must appear at the top ranks. This metric is computed using the question, ground truth answer, and the contexts, with values ranging between 0 and 1, where higher scores indicate better precision.
- **Faithfulness** measures the factual consistency of the generated answer against the given context. It is calculated from the answer and retrieved context. The answer is scaled to the (0,1) range. Higher the better. The generated answer is regarded as faithful if all the claims that are made in the answer can be inferred from the given context. To

calculate this a set of claims from the generated answer is first identified. Then each one of these claims is cross-checked with a given context to determine if it can be inferred from the given context or not.

- **Answer Relevancy** - this evaluation metric focuses on assessing how pertinent the generated answer is to the given prompt. A lower score is assigned to answers that are incomplete or contain redundant information and higher scores indicate better relevancy. This metric is computed using the question, the context, and the answer. The Answer Relevancy is defined as the mean cosine similarity of the original question to several artificial questions, which were generated (reverse-engineered) based on the answer.
- **Context Recall** - it measures the extent to which the retrieved context aligns with the annotated answer, treated as the ground truth. It is computed based on the ground truth answer and the retrieved context, and the values range between 0 and 1, with higher values indicating better performance. In an ideal scenario, all sentences in the ground truth answer should be attributable to the retrieved context

By evaluating the model, we can say we achieved a good performance on the first dataset Universal Declaration of Human Rights reaching a context precision level of 90%, as well as a faithfulness score of 94%. Regarding answer relevancy, we could achieve 85% and a context recall score of 90%. Overall, on the first dataset, we achieved satisfactory results. Moving to Magna Carta the results showed less impressive results: we achieved a faithfulness level of 25.6% and answer relevancy of 32%. The reason behind this could be the Magna Carta, the specific old English style in which the LLM model could not capture the structure of the sentences, and some words are not used in modern English.