RAG System Specialized for Tabular Documents

Development Track

Group 39 - Hongxiao Chen(hc39@illinois.edu)

[Functions and Users]

This project plans to develop a software system that specializes in extracting table data from documents and later provides responses based on these data for user queries. This project is an extension to retrieval-augmented generation system where a large language model is included to provide responses that is highly related to existing sets of documents. This system is designed for companies that keep a large storage of documents mixed with text reports and a large number of tables. Through this system, they can accurately extract their desired information without opening each document and checking every table.

[Significance]

This system seems trivial at first glance, since large language models are already performing well in extracting information from pure textual documents. However, for data inside tables, there is still a significant issue of illusion where the model may report a completely irrelevant row or column when we ask for specific value. For example, there is a report of products sold at a store in 2024, where each table contains the sales in each month. If we ask the model to provide the sales information of specific product in February, it may give irrelevant information on another row of tables.

People may think that this can be addressed by searching in a database system, but for employees who work on advertisement that do not know SQL language, querying in database may be difficult, and they might need to request help from another department. If there is a reliable RAG system that extracts information from tables inside the reports, it may improve the productivity of the company. The author meets this problem while having an internship in a company and wishes to address it by providing an integrated system that does not require knowledge in computer science.

[Approach]

The author plans to build a system that first takes documents and vectorize them. Note that for tabular data, some key information might not be inside the table, to improve the precision of searching tables, the author might try to add “tags” to tables, like their titles and headers. The context of the table could also help to understand the table. The author will also consider the TaBERT model, which can extract and encode table from natural language context.

After embedding raw texts and tables, they will be stored in a vector database for fast retrieval. The author is familiar with FAISS but also willing to try Pinecone or Chroma. Note that for user query that requires comparison between multiple tables, the “top-k” parameter should be adjustable so that information will not be lost.

Finally, an LLM serves for taking user input, convert it into search query, find the desired information from top-k documents retrieved from the vector database, and convert it into natural language response with reference to the original file for easier confirmation. Since

company data are secrets, the LLM should be deployed locally to avoid leaking. Based on the limitation of computation power and storage space, the author will try small models like GLM4-9B, Meta-Llama-3-8B, and DeepSeek-R1-Distill-Llama-8B to find the best model for extracting and concluding tabular information.

[Evaluation]
The author will find documents with mixed texts and tables and test each model. This includes basic data finding (like the number of apples sold in March 2024), computing in a table (total sale price in March 2024), and cross table information retrieval (which month has the highest number of apples sold in 2024). Based on the correctness and response time of these queries, the best model is decided. The author will also test these standards on tagged tables and tables with contexts.

[Timeline]
First week: Download models, try local deployment, write some test documents.

Second week: Implement document segmentation, try TaBERT for embedding tables, try tagging and including context for tables.

Third week: Implement FAISS/other tool for vector database retrieval, try including LLM for converting user queries to FAISS search queries.

Fourth/Fifth week: Test different models and embedding methods on test data, compare results, write reports, make a UI for users to import documents and search.

[Task division]
Hongxiao Chen – Everything.