Retrieval Augmented Generation with NYC Building Data

Cesar Herrera
CUNY Graduate Center, New York, United States
Email: cherrera@gradcenter.cuny.edu

Abstract

This paper will focus on a practical application using a Retrieval Augmentation Generation model based on New York City building TR-6 documents to extract tabular data and identify key characteristics for building code classifications. This will build a retrieval data repository containing the TR-6 documents that are used to predict if a building is considered SAFE, UNSAFE, or SWARMP (safe with repair and maintenance program), but then followed up with using Large Language Model label predictions. This will extract text to review key characteristics the RAG identifies in classifying the document. Future analysis will be done on the key features extracted for each building.

1 Introduction

The DoB, Department of Buildings has a program called FISP, Façade Inspection Safety Program, that reviews the conditions of buildings. In the TR6 reports submitted to the DoB, building Façades can be classified as either Safe, SWARMP (Safe with A Repair and Maintenace Program), or Unsafe. All buildings that contain 6 floors or above after the initial five years the building can be occupied must perform an inspection. The inspection must be conducted every 6 years after. The DoB requested a RFP (Request for Proposal to evaluate the FISP program. Part of this proposal includes looking at hundreds of these TR6 documents. The DoB deliverable is a report identifying key characteristics for buildings that are failing the FISP program. DEliverables also include a presentation of any findings that can help improve the FISP program for the city.

It is time consuming to review each document one at a time by an architect. Therefore, it is beneficial to build a Retrieval Augmented Generation model to improve the process of data extraction for then human review. The Retrieval part of the project breaks each TR-6 document down to sets of characters, storing them in a data repository in a vectorized form. Then when building the prompt for the Large Language model(LLM), the retriever will query the data repository to extract relevant passages to pass the Large Language Model. The Generative portion combines the retriever prompt and a question or instruction to the Large Language model for a response. For the data extraction, the response is stored in a tabular format. The retrieval prompts is printed and can be passed to a text file for further analyze.

2 Overall Architecture

Figure 1 is a diagram of the general architecture of a RAG model. The process begins with the blue rectangle. Here the input documents are the TR-6 documents. The code written in python uses langchain to split the PDF document using a text splitter. The split text is vectorized using the Hugging Face Encoder library to improve the performance of the RAG, leveraging vector math to make the comparisons to retrieve relevant text segments. The text is stored in a FAISS vector database.

The lower square builds the retrieval prompt and user prompt(the question or instruction for the LLM). Part of the prompt retrieves the relevant text from the FAISS database using the Hugging Face encoder to extract the relevant passages. It returns the top K relevant chunks as context for the users question. This context is combined with the user question to the large language model using Google's Gemini genai. Gemini then provides a response.

3 Named Entity Recognition

To extraction specific information using the Retrieval Augmented Generation, the prompts need to define the definitions of the values to search. The task of extracting name entities or predefined

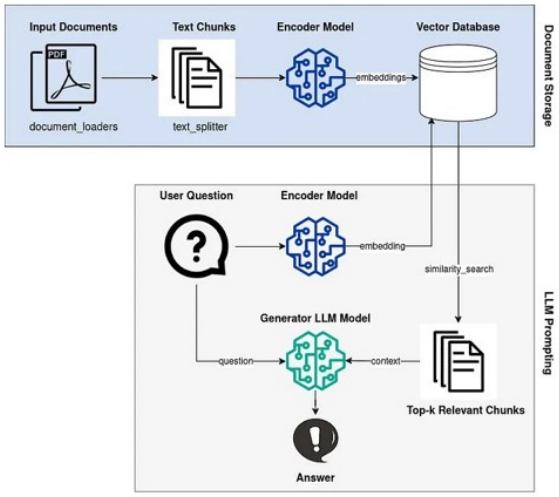

Figure 1: RAG Architecture

categories from unstructured data is called Name Entity Recognition. By stating what to extract without name entity context reduces the desired outcome because of ambiguity or cause large language model hallucinations. For instance, when attempting to extract the building identification number or BIN, genai fails to extract the content. This is because it identifies BIN as bank identification number and fails to extract because of privacy protections. An example of hallucination is extracting data such as BIN or Control number without prompt context, genai would confuse the values and swap for any number found in the context window.

> prompt: "Return a csv format separated by "|" as a delimiter for building address, borough, building identification number, control number, Exterior wall type, Exterior wall materials. The building identification number or BIN is a 7 digit number starting with numbers 1-5. The control number is a six digit number that starts with the cycle number 9. Make sure to format in to six columns"

By defining the name entities to extract in the genai prompt, this improves the probability of success in data extraction. After the relevant chunks are extracted, the above prompt was passed with the data to improve the chances of extracting the right data. This provided genai with the definitions to extract the correct content and not set off the privacy concern. Out of 450 documents, the application was successful at extracting data from 408 documents. When reviewing the documents that failed using the Named Entity Recognition prompt, some of the categories for "Exterior Wall Type", or "Exterior Wall Material" were very long or empty. The program had a 90.6% success rate and extracting the PDF content. The other reports can be done manually.

4 RAG Model Document Prediction

In addition to extracting tabular data from the PDF documents, the Retrieval Augmented Generation was used to predict the building classifications. To compare results, first the program would attempt to predict each document isolated. Meaning only one document is stored in the FAISS database to pass to the genai. Then the second part of the experiment stores 100 documents in the database to assist with future document predictions. The program would then store each document and come up with a prediction of the building classification.

> Safe: 80 Reports, 28.5%
> SWARMP: 152 Reports, 54.1%
> Unsafe: 49 Reports, 17.4%

The reports available to use for the program is not evenly distributed. The SWARMP classification is heavily represented with over half the reports in this classification. The Unsafe is least represented with less than 1 in 5 documents in this classification. One of the methods to improve the prerequisites in this experiment would be to acquire an equal number of reports in each classification, but since there is a small number of reports to start with, all the reports available were used.

To reduce the probability of data leakage, each document after the initial 100 would need to redact the classification. Before storing the document in the FAISS vector database, the document would need to remove the SAFE, SWARMP, and UNSAFE words before getting vectorized and stored. This reduces the data leakage problem.

4.1 Single Document Retrieval

When running the application with only a single document the program mislabeled approximately 1/3 of the data. This was mostly misrepresenting the SWARMP documents as SAFE. After examining the retrieved text that was passed for some of these labels, the context prompt provided samples that regarded the building as satisfactory, or repairs that fix the building conditions. In this sense the

Actual \Predicted Labels	SAFE	SWARMP	UNSAFE
Single Document Retrieval			
SAFE	65	11	4
SWARMP	77	65	10
UNSAFE	2	4	43
Historical Retrieval			
SAFE	70	25	2
SWARMP	33	94	2
UNSAFE	1	11	42

Table 1: Rag Model Prediction Matrix

Labels	Precision	Recall	F1-Score
Single Document Retrieval			
SAFE	0.45	0.81	0.58
SWARMP	0.81	0.43	0.56
UNSAFE	0.75	0.88	0.81
Historical Retrieval			
SAFE	0.67	0.72	0.70
SWARMP	0.72	0.73	0.73
UNSAFE	0.91	0.78	0.84

Table 2: Rag Model Performance Matrix

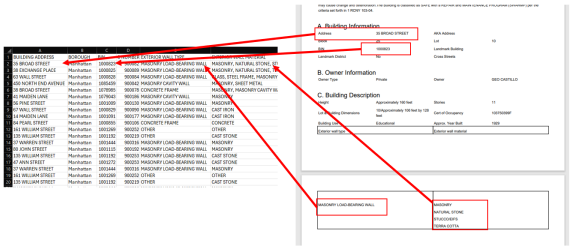

Figure 2: Named Entity Recognition Example

positive language could have had more weight in mislabeling the SWARMP documents. Observing the outputs of tables 1 and 2, SWARMP had a high value for precision but low recall score. Out of all the documents labeled SWARMP most of these were labelled correctly, mostly because most of the documents are SWARMP. But for all the documents that were SWARMP that were mislabeled, more than half were incorrect, leading to a low recall score. This was the opposite for SAFE but for the same reason that many SWARMP documents were mislabeled SAFE. The UNSAFE documents preformed well in all performance metrics. Looking at the retrieved text, this language was less ambiguous. Keywords such as failed, unsatisfactory, urgent work, file extension were present in findings.

4.2 Historical Retrieval

Building a database containing a base of 100 PDF documents improved at predicting document label. While the precision of the SWARMP and recall of the SAFE documents decreased, the precision of SAFE and recall of SWARMP improved. It appeared the SWARMP retrieval was more conservative at passing positive context words that can mislabel as SAFE. By providing more documents that are SWARMP, more of the retrieved context will align with a SWARMP based classification. Making the application lean towards SWARMP as a prediction. Looking at table 1, shows the number of predicted SWARMP labels increased in comparison to the single document retrieval application. This influenced a bias in the retriever part of the database.

4.3 Techniques to improve RAG

There are multiple approaches that can be used to improve the application on the retrieval architecture or context chunks passed from the retriever. These approaches include MultiQuery retrieval, contextual compressor, and ensemble retrieval. MultiQuery retrieval uses multiple questions or queries to provide a larger context for the retrieval prompt. Similar to how the query in the section 3 data extraction provided definitions and context to im-

prove the large language model, this approach involves doing the same for the retrieval prompt. The second approach is contextual compressor, this can improve the application for multiple reasons. When retrieving the chunks, the application can reduce the embeddings that are relevant to the prompt. This can reduce the noise passed to the large language model. Also, because the chunks are compressed, the application can increase the number of chunks retrieved. This benefit is related to the character size limit Large Language models have for their prompt window. For genai, the character limit is 3200. The ensemble retrieval approach uses multiple retrievers with different weight assignments to provide relevant documents according to the assigned weights. For the application, if multiple prompts focused on specific conditions of the building such as replacement, repair, and appurtenances conditions, providing a broader related context of the building profile.

5 Conclusion

The Retrieval Augmented Generation application successfully extracted tabular data from different sections of the PDF. Most of the tuning for the architecture is done on the retriever side. A large language models is huge and cannot be tuned in a single processor or personal computer. What can be improved is the prompt engineering aspect of the RAG; this includes the chunks passed in the architecture. Simple items that can be tuned are the chunk sizes and the number of relevant chunks queried. More complex tuning approach techniques are MultiQuery retrieval, contextual compressor, and ensemble retrieval. These techniques also focus on how the chunks retrieved are extracted. Future work would include using this application to query a larger set of DoB PDFs to construct tabular table to perform further analysis of the DoB FISP program.

6 References

References

Pavlos Vougiouklis Nikos Papasarantopoulos Jeff Z. Pan Wenyu Huang, Mirella Lapata. 2023. Retrieval augmented generation with rich answer encoding. *Anthology*.

Yile Wang Peng Li Zhicheng Guo, Sijie Cheng and Yang Liu. 2023. Prompt-guided retrieval augmentation for non-knowledge-intensive tasks. *Anthology*.