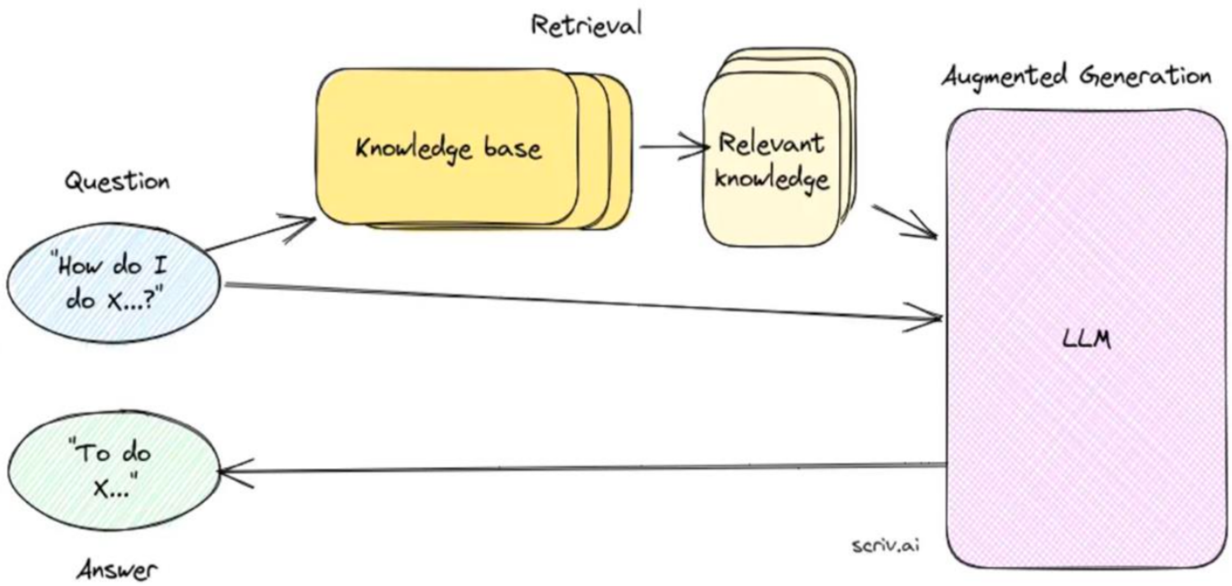
What is RAG: -



First, I used PyPDFLoader to load the pdf and we then break them into different pages and these pages are then split into documents using RecursiveTextSplitter, then these documents are embedded and then stored in a vector database. Then the vector database is queried against the vector database which returns the top-K relevant documents, then the question along with the retrieved documents are given to the LLM for generation.

Details regarding the project: -

Initially, when I received the take-home project, my first idea was to design the pipeline so it could work with any PDF document. The document would be embedded, stored in a vector database, and the database would be used as a retriever. It would query relevant documents and pass them to the LLM for generation. I chose LangChain as my framework because it is user-friendly and offers many excellent integrations. Additionally, I appreciate LangChain for its comprehensive documentation.

A) How did I construct the dataset: - For creating the dataset I used Ragas test generator initially I used different distributions for {simple, reasoning, multi-context queries}. Did some initial testing using {0.25, 0.5, 0.25} respectively to see whether the model works well or not, but the simple questions were very straight forward, and I was getting {'faithfulness': 0.9246, 'answer_relevancy': 0.9458 [as shown in the last page of this document]}. But I wanted my dataset to consist of but more complex queries hence I used {0, 0.75, 0.25} distribution and along with them I also added few simple questions to the data which I think are some of the question users might want to ask. And also the queries does not belong to one page or any topic but it is diverse and consists of different questions or queries from the user's perspective.

B) How and why, you chose these evaluation metrics: - While researching and learning about RAG and while building projects I used to get and used to see a lot of questions on the internet that, how sure are you that the answer generated is not due to hallucinations etc. During this time, I learnt about Ragas, this is a very easy to use library for evaluation the outputs generated by the model.

The two metrics that I choose and think important are Faithfulness and Answer relevancy

1) Faithfulness: - It evaluates whether the LLM, is outputting the answers which are factual and does not hallucinate and contradict any information that is not in the context provided to the LLM. The reason I choose that the answers are reliable and correct.

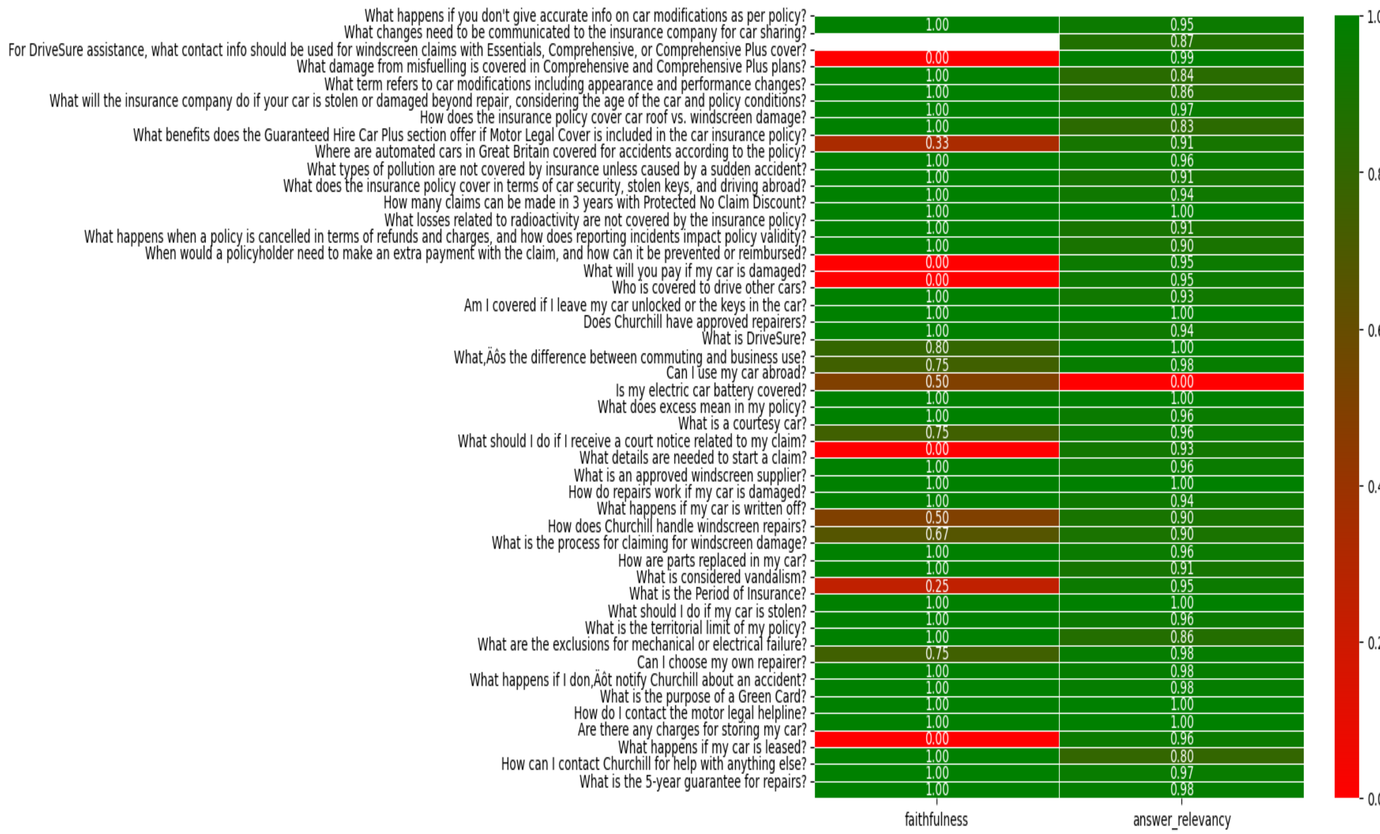
2) Answer relevancy: - It evaluates how the output is relevant to the input that is given to the LLM. This metric assesses if the instructions given to the language model through the prompt template result in generating relevant and useful outputs, considering the provided context for retrieval.

C) What did you try to improve the accuracy: -

Different methods I tried to improve the model: -

1) RetrievalQA Chain from LangChain
2) Multi-Query Approach
3) HyDE (Hypothetical Document Embeddings)
4) Re-Ranking
5) ColBERT (Contextualized Late Interaction over BERT)

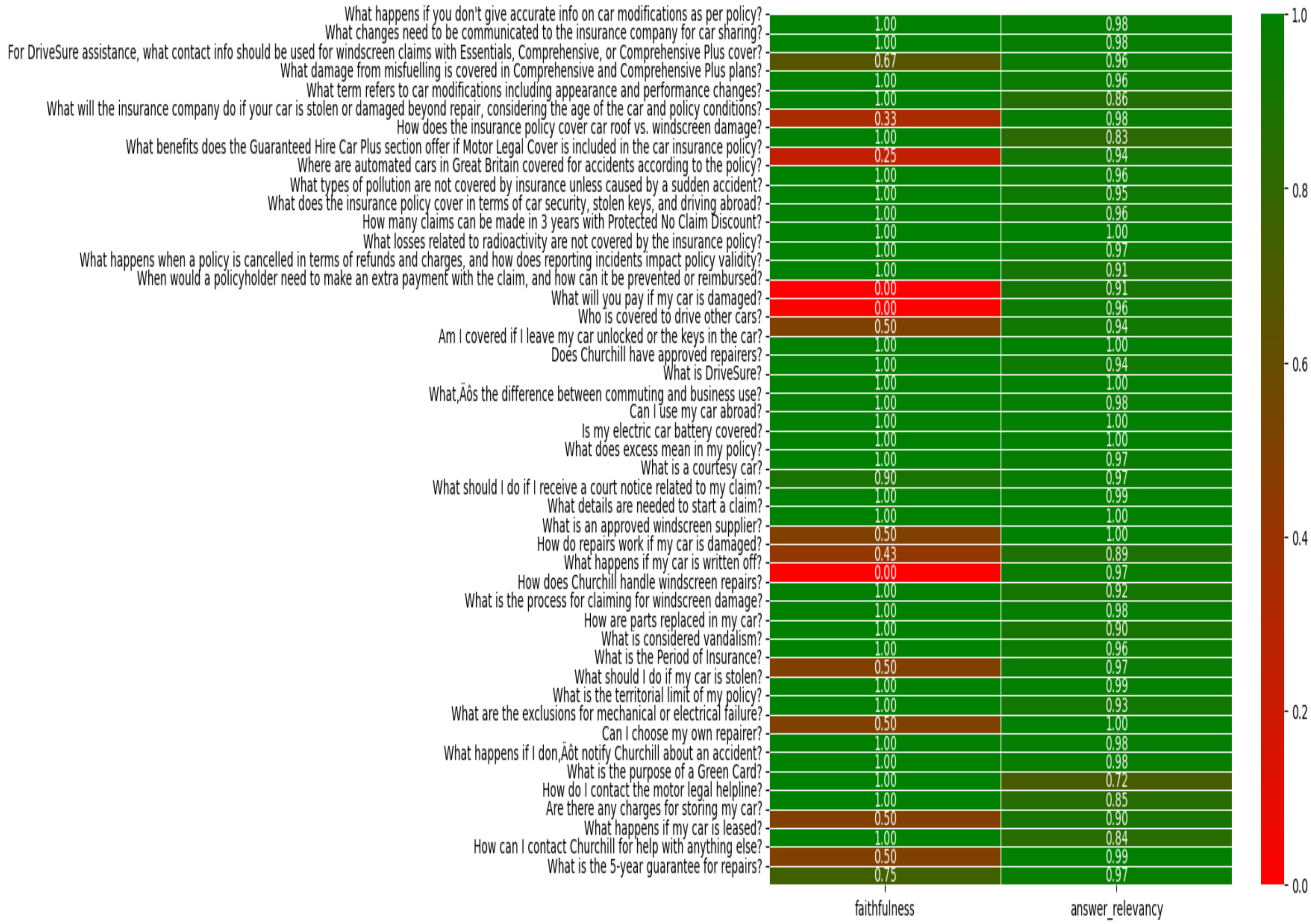
RetrievalQA chain: - Given a user query this chain will query the the retriever which is the database that consists of the embeddings of the data / document we would like to query. The below are the results using the simple RetrievalQA chain.



{'faithfulness': 0.8067, 'answer_relevancy': 0.9225}

Multi-Query Approach: - This is an advanced RAG technique, most of the time user queries are ambiguous and they tend to also receive an incorrect/ambiguous answer. To mitigate this problem in this approach I used a llm to generate 5 questions which are similar each other. The vector database is then queried with all the questions and retrieves documents and now we take the union of all documents and pass the question and all the documents to the LLM for generation.

Why do we do this? So, we can capture more similar documents to the question and give more context so we can give more accurate answer to the user.

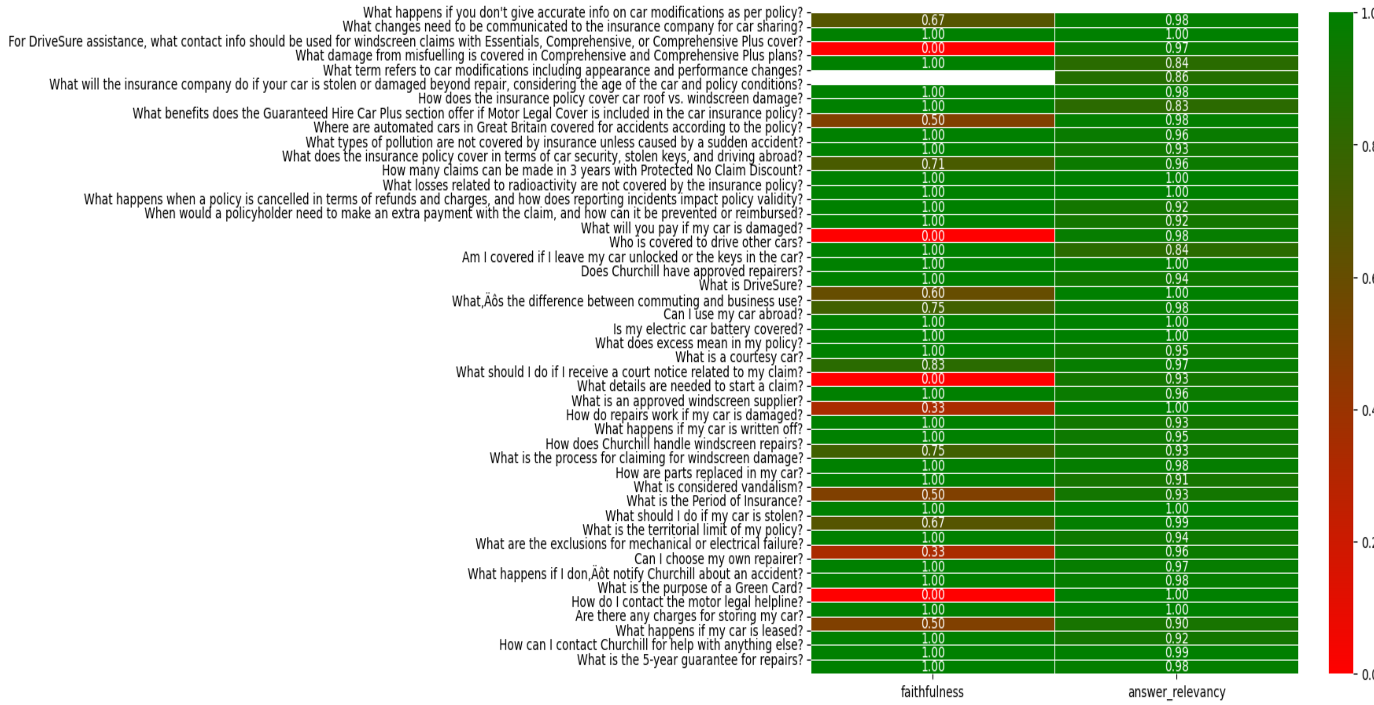


{'faithfulness': 0.8115, 'answer_relevancy': 0.9487}

HyDE(Hypothetical document embeddings) :- In this process we take the query, we use an LLM to generate a hypothetical document(we want the llm to hallucinate) and then the vector database is queried against the document and the query. In theory this has improved the performance of RAG system but in my system the results are actually very bad.

{'faithfulness': 0.4553, 'answer_relevancy': 0.4708}

Re-Ranking: - In this process like multi-query we generate 4 queries and the vecator database is queried against the 4 queries and all the generated documents are then passed through a reciprocal rank function which ranks the documents and returns them in descending order. As expected, the answer relevancy has increased because the ranking takes place after retrieval. [used the langchain reciprocal rank fucntion]



{'faithfulness': 0.8033, 'answer_relevancy': 0.9552}

RAG with colbert:- This is a new technique compared to the other above in this we download a checkpoint of the colbert model. Instead of taking the document and embedding them, we take the documents and break them down into tokens and we embed the tokens rather the document and similarly we do the same thing for the question. In every token in the question, now we are comparing the similarity with every token in the document. The final score is the sum of similarities between every token in the question to any token in the document. This is the method I am choosing for my final model, even though there will be some latency, but the faithfulness of the model is very high compared to any of the approaches I tried and also the answer relevancy is good enough.

{'faithfulness': 0.8980, 'answer_relevancy': 0.9082}

The below is just an example I tried using {0.25, 0.5, 0.25} distributions for my test data using Ragas, I used the RetrievalQA chain.

{'faithfulness': 0.9246, 'answer_relevancy': 0.9458}

Reference:

1) https://python.langchain.com/v0.1/docs/get_started/introduction
2) https://docs.ragas.io/en/stable/index.html
3) Gao, Luyu, et al. "Precise zero-shot dense retrieval without relevance labels." *arXiv preprint arXiv:2212.10496* (2022).