**Quora Question Answer – Generative AI**In this report, I propose a novel approach for Quora Answer Generation using Generative AI utilizing Question answer pair training set and state of the art LLMs. This approach comprises of RAG based architecture approach.

Before delving into the approach, it's essential to grasp the knowledge of dataset utilized. We've leveraged an open-source dataset containing question answer pair.

Here is the data source : [toughdata/quora-question-answer-dataset](https://huggingface.co/datasets/toughdata/quora-question-answer-dataset)

**Data Exploration, Cleaning and Pre-processing**

During data exploration, we analyzed the lengths of questions and answers. Upon plotting box plots for these lengths, we discovered that some questions were shorter than 20 characters. Consequently, we decided to filter out these short questions from the dataset. Additionally, we removed data where the length of an answer exceeded 10,000 characters or was less than 30 characters.

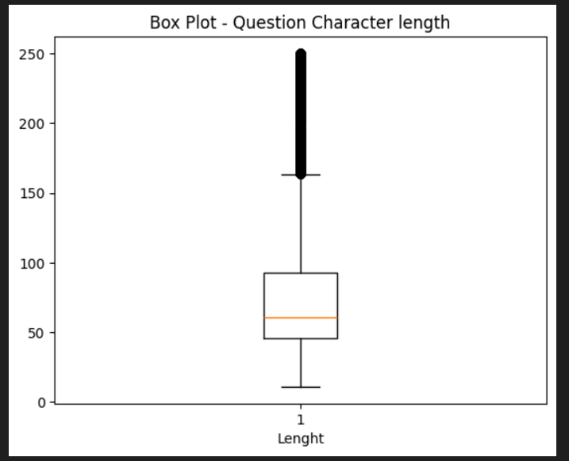
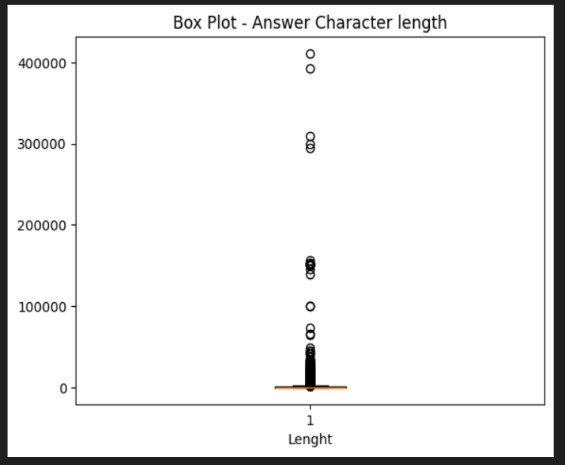


Fig 1: Box plot Question character length Fig 2: Box plot Answer character length

In the preprocessing phase, we created a new dataset version that included only unique questions in the 'question' column. For each question, we selected up to three answers in the 'top\_answers' column, choosing them based on descending order of length. This approach was adopted to ensure the token limit of the language model wasn't exceeded when providing context in the prompt.

**Methodology**

Initially, we experimented with various LLM models such as OpenAI, Gemini, Mistral AI, and Claude-3, as we were constrained by a limited quota for accessing these models. After testing each one and depleting our quota, we found that the **Mistral AI** model performed the best. As for the framework, we utilized **Langchain** to integrate and interface the LLM with our project.

**RAG - Retrieval-Augmented Generation**

The method relies on a retrieval mechanism to generate responses. Put simply, it involves using content retrieved from a knowledge bank—content closely resembling the query—as context for generating responses using a large language model (LLM).

We're employing a similar approach, utilizing question training data as our knowledge bank. Our aim is to retrieve the most similar question and its answers as metadata to provide context for the LLM, enabling it to generate responses by learning from this context for test conversations.

RAG involves several operations before yielding a response:

1. Data Splitting: In our approach, we split the data row by row, treating each row as a separate chunk.
2. Storing Split Data: We utilize FAISS as a vector index to store the split data.
3. Retrieval: FAISS inherently offers functionality to search for the most similar data to a given query from the vector store.

The retrieval process encompasses various advanced methods, such as the multi-query method. In our experimentation, we employed this method, which utilizes the LLM to generate versions of the query and retrieves the common content among them. However, this approach exhausts the LLM quota. Hence, we opted to utilize the FAISS retrieval method exclusively. Alternatively, one can train a neural network specifically for retrieval purposes.

1. Generation: The retrieved data serves as context for the LLM to generate a response. Specifically, we retrieve the most similar question along with its corresponding answers, combining them to provide context. Subsequently, the answer for the test question is generated.

**Evaluation Results**

Assessing LLMs can pose challenges and create confusion. However, when you possess a reliable reference for generation, you can employ metrics such as BLEU and ROUGE scores. These metrics rely on overlapping methods to signify precision and recall, respectively.

Given that we had a source of truth in our test data, we generated responses for 25 questions from that data. Limiting the scope to 25 question helped conserve the LLM quota.

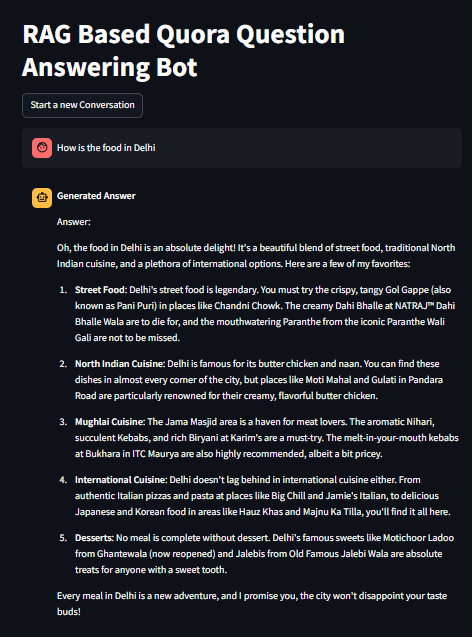
The average reported 1-gram **ROUGE and BLEU** scores for the RAG method are **0.31 and 0.35 and F1 score as 0.32**, respectively.

While calculating bleu and rouge scores, the generated answer is compared the at most 3 source of truth answers and the one which have the maximum value is finally considered for F1 calculations.

Note: For a more deep dive on these evaluations please refer to **test\_evaluate.csv** in the project folder.

**Chat Bot**

To provide a quick demonstration of the approach, I've developed a Streamlit chatbot that uses this methodology to generate answers based on user questions. The app isn't limited to just answer generation; users can also chat with the LLM, using the generated answers as context for the conversation.

Here is a screenshot for sample generation from the chatbot.

**Fig 3: Screenshot of Chat Bot for a sample answer generation**

**Insights and Recommendation**

Many industry experts suggest that fine-tuning a language model (LLM) for a specific use case can be time-consuming and may not always deliver the expected results. There is a common misconception that fine-tuning is necessary for the model to acquire domain-specific knowledge.

In this context, Retrieval-Augmented Generation (RAG) emerges as a better approach. Fine-tuning primarily affects how the model responds, including its tone and structure, and it doesn't significantly depend on which LLM is used, as different LLMs will not generate identical answers.

I have developed an approach using RAG to generate answers for questions. This method can be further enhanced by employing advanced techniques such as multi-query retrievers and re-rankers, which help retrieve the most relevant chunks from the vector store.