IIT BHILAI

COMPUTER SYSTEM DESIGN

Final Phase: Project Report for
"AI Chatbot for Efficient Network Troubleshooting"

Submitted to: Dr. Gagan Raj Gupta
Associate Professor

Team Diagonals:

Moyank Giri	12310830	M.Tech DSAI
Harshit Kumar	12310680	M.Tech DSAI
Tanmoy Bhowmick	12311110	Mtech DSAI
Manish Rai	12310790	Mtech DSAI

Contents

1 Motivation

The motivation for our project stems from the critical role that base stations play within our network architecture, acting as pivotal points for sending and receiving messages in accordance with protocols and standards established by international and standard bodies such as 3GPP and IETF. Given the vast array of knowledge and specifications required for troubleshooting networks, network administrators often face significant challenges. Manual troubleshooting processes are not only time-consuming but also prone to inefficiencies, leading to delays in resolving network issues. To address this challenge, we propose the development of a network-based chatbot. This innovative solution aims to streamline the troubleshooting process by providing instant access to relevant information and guidance, thereby enhancing the efficiency and effectiveness of network administrators in maintaining and optimizing network performance.

For the Final Phase, this document is updated with details of the final implemented solution and the technologies used, including the entire pipeline for the Advanced RAG Retreival-based LLM Model starting from Embedding Generation to LLM Integration along with its User interface and public deployment.

2 Business Objective

Develop an AI-powered chatbot designed to assist network administrators, support staff, and users by accurately understanding queries related to network issues and providing immediate, conversational responses. The chatbot will leverage advanced NLP techniques and the Retrieval-Augmented Generation (RAG) approach to not only offer precise answers but also suggest relevant documents for in-depth solutions. It will focus on common network problems, utilizing a Vector Database for document embeddings to enhance the accuracy and efficiency of the retrieval process. This system aims to improve the troubleshooting experience, reduce resolution time, and facilitate access to comprehensive information, thereby increasing operational efficiency and user satisfaction in network management contexts.

3 Literature Study

The following section contains details of methodologies relevant to the proposal which includes Proxy Tuning, RAG Fine Tuning, VectorDB, etc from papers and articles.

3.1 Proxy Tuning

Proxy-tuning addresses the challenge of efficiently customizing large pre-trained language models (LMs) to better suit specific needs and tasks. Proxy-tuning introduces a lightweight decoding time algorithm that operates on top of black-box LMs.
The proposed approach involves the utilization of a smaller language model (LM), denoted as the expert (M+), fine-tuned on a specific task, alongside an anti-expert (M-) derived from the original untuned version of the smaller LM. During decoding, a logit offset is computed at each step based on the disparity between the experts' and anti-experts' predictions. This offset is then added to the larger pre-trained LM (M) logits, guiding its predictions. The logit offset shifts the base model's predictions, effectively aligning them with the expertise learned by the smaller tuned LM. This decoding-time logit offset mechanism facilitates the generation of text that adheres to the desired behavior or knowledge acquired by the expert, enhancing the base model's output in alignment with specific tasks or domains.[1]

3.2 LLM Fine Tuning with RAG

The innovative approach coined RA-DIT (Retrieval Augmented Dual Instruction Tuning), presents a novel adaptation of the RAG framework. In RA-DIT, the RAG dataset, comprising queries, retrieved-context, and responses, is leveraged for fine-tuning a large language model (LLM). This fine-tuning process utilizes RAG outputs as training data, fostering improved LLM comprehension of contextual nuances. During fine-tuning, when presented with a query, a retriever identifies relevant context from the RAG database, and both the query and context are subsequently provided to the LLM for enhanced interpretation. This integrated process optimizes the LLM's performance by aligning it closely with the specific contextual requirements defined by the RAG dataset.[2]

3.3 VectorDB

A vector database is a collection of data stored as mathematical representations. Vector databases make it easier for machine learning models to remember previous inputs, allowing machine learning to be used to power search, recommendations, and text generation use cases. Data can be identified based on similarity metrics instead of exact matches, making it possible for an ML model to understand data contextually.
The proposed methodology creates embeddings of the collected data and stores it in VectorDB enabling fast semantic search in the database. [3]

4 Proposed Methodology

The following section entails details about the proposed methodology, including details such as Data Collection, ML Objectives and Models, High-level design, Metrics, etc.

4.1 High Level Design

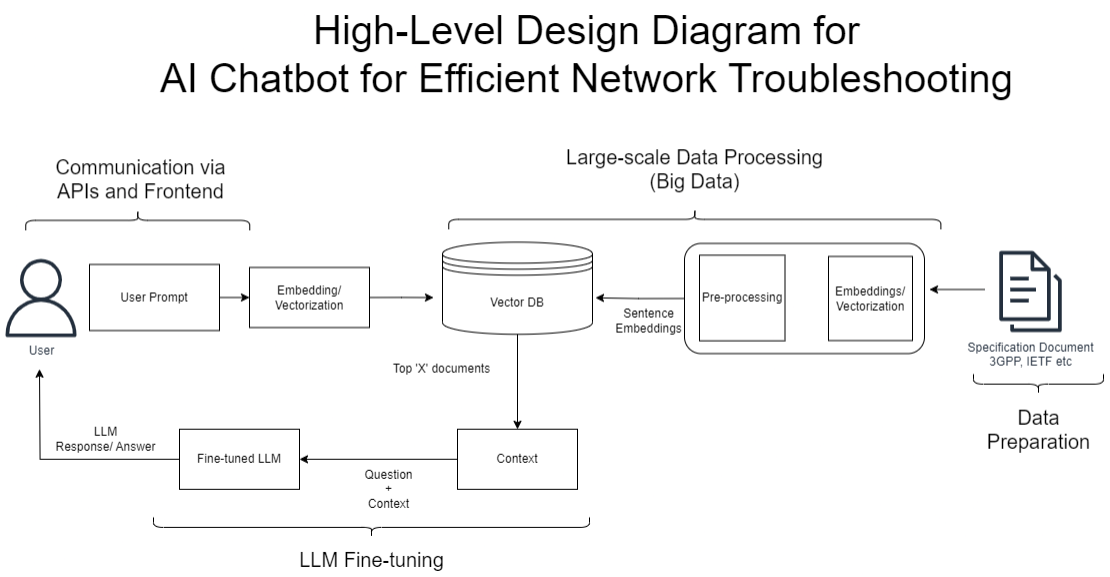

Figure 1: High-Level Design Diagram

As it can be seen in Figure 1, there are 4 major modules in the project which include

- APIs and Frontend Communication
- Large Language Models (LLM) Fine Tuning
- Data Preparation
- Large-Scale Data Processing (Big Data) and Vector Database

The project flow starts on the user end where the user passes a question as a prompt. This prompt is converted to an embedding vector, which is used to find the documents in the VectorDB that are the most similar to the query. The VectorDB is initially populated with vector embeddings of various international standards, using Big Data Technologies, enabling the finding of the most suitable response for the question. This set of similar documents is called the "context" vector, which is then passed along with the question to the Fine-tuned LLM, from which the response is generated.

4.2 Data Collection

The proposed methodology intends to curate a dataset in CSV format for the purpose of answer generation. This dataset will encompass questions, corresponding answers, and the contextual information associated with each answer. Simultaneously, a separate dataset tailored for retrieval tasks will be compiled. This dataset will consist of questions, accompanied by their respective sections and subsections.

The primary source for data collection will be international standards organizations renowned for their contributions to mobile telecommunications protocols. Key entities in this context include 3GPP, IETF, and ETSI. Through meticulous extraction and organization, these datasets will serve as valuable resources for advancing answer generation and retrieval tasks within the domain of mobile telecommunications standards

4.3 ML Objective and Models

The ML objective of the proposed methodology mainly revolves around 2 tasks: embedding generation and query-response generation. For the task of embedding generation, the ML objective is to minimize the loss of the hypothesis function such that it generates efficient representations of the input texts. For the query-response generation task, the ML objective is to maximize/minimize the score/metrics used such that the response generated from the chatbot is optimal and correct for any query the user gives.

For the core of the chatbot in the proposed methodology, Mixtral 8x7B, an open-source Large Language Model (LLM) has been chosen, due to its superior performance and efficiency that surpasses GPT-3.5 and Llama 2 70B in benchmarks, offers instructive excellence with 5x fewer parameters during inference for reduced compute costs, and ensures fine-tuning flexibility. We will integrate an embedding model that aligns with Mixtral AI's architecture, optimizing our retrieval process. By applying the Retrieval-Augmented Generation (RAG) model, we aim to seamlessly blend generative capabilities with targeted document retrieval, ensuring our chatbot delivers precise answers and relevant document suggestions efficiently.

4.4 Metrics

Large language models (LLMs) are evaluated based on various metrics to assess their performance and quality. Common metrics include perplexity, which measures how well the model predicts a sample text; BLEU (Bilingual Evaluation Understudy), often used to measure the similarity between the generated text and a reference text; ROUGE (Recall-Oriented Understudy for Gisting Evaluation), which assesses the quality of summaries or generated text by comparing them with human-written reference summaries, etc. These metrics serve as valuable tools for quantitatively evaluating the performance of language models across various natural language generation tasks.

5 Implementation Details

This section entails details of the Implementation which include the current progress of the proposed implementation, the methodologies employed, and the tools used for various tasks such as Data Collection, Generation of Vector Representations, VectorDB Implementation, LLM Fine-tuning and Deployment.

5.1 Data Collection

For the project, focus was placed on creating a dataset of around 500 questions was sourced from 3GPP documents along with data extracted from community forums such as Quora, Reddit, StackOverflow, and StackExchange. Furthermore, a telecom-specific dataset is also retrieved, which consists of 10,000 MCQ-type questions. This dataset serves as a foundation for fine-tuning Language Models (LLMs). Organized into distinct categories to facilitate the training and refinement of LLMs, the dataset is divided into three main categories: General Telecom users, representing discussions and queries from platforms like Reddit and Quora; Academic Researchers, encompassing scholarly discussions typically found among students from institutions like IIT; and Telecom Developers, comprising technical queries and discussions primarily sourced from StackOverflow and StackExchange. This categorization strategy enables precise targeting of data for LLM fine-tuning, ensuring that the resulting models are adept at understanding and generating content relevant to these specific user groups within the telecommunications domain.

5.2 Generation of Vector Representations

The data is collected from various sources as described in subsection 5.1. All these documents need to be stored in a Vector Database for efficient search, and therefore, the project requires efficient production of vectors for any input data which is compatible with the underlying LLM model. The project currently employs the SFR-Embedding-Mistral and all-MiniLM-L6-v2 embedding model provided under HuggingFace. These embedding models have been chosen as they provide a good balance between speed and performance while also being very compatible with the underlying Mistal LLM. Before passing the input documents to the model, partitions/chunks are created for the documents as per the maximum tokens allowable by the embedding model. Once the chunks are created, individual embeddings, each of size 4096, are created from the model for each chunk. These embeddings/ vector representations are then stored in the Vector Database

5.3 VectorDB Implementation

The choice of vector database for the project is Qdrant VectorDB. Qdrant was the choice of VectorDB as it has been shown to achieve the highest Requests-per-Second (RPS) and lowest latencies in almost all scenarios, regardless of the precision threshold and metric chosen. On the Vector database, the metric

chosen for similarity search is the cosine similarity. The main reason for choosing cosine similarity is its scale-invariant nature and using angle measurement for similarity, which provides a more intuitive sense of similarity. Using this vector database, the plan is to extract a "context" for the input prompt from users. This context will comprise some of the top documents similar to the input prompt, allowing the LLM to provide the users not only the answer to their prompt but also to provide the source of the answer provided by the LLM. The implementation included using content as well as questions for context generation, which is to be passed on to the LLM.

5.4 User Interface Implementation

Display bugs observed across various screen sizes have been addressed to enhance user experience and ensure consistent presentation. Furthermore, a download feature has been integrated, allowing users to retrieve answers to their queries and provide feedback conveniently. This comprehensive setup not only guarantees smooth functioning but also prioritizes security and user satisfaction. With these implementations, the chatbot is poised to deliver reliable and accessible assistance to users while maintaining a secure online environment.

5.5 Deployment

The deployment of a chatbot on the public server, provided by the ITIS office under radomchatbot.iitbhilai.ac.in, has been accomplished. Apache serves as the reverse proxy server, efficiently managing incoming requests. To ensure secure communication, a certification chain has been established for the website, utilizing Certbot for automated renewal of certificates.

6 Results

This section describes details of the obtained results from the aforementioned tasks

6.1 VectorDB Docker Environment

The vector DB environment is provided as a docker container available on port 6333 and 6334 as shown in Figure 2

```
moyankgiri@s3-workstation:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND             CREATED         STATUS         PORTS
                  NAMES
6384cae01373   qdrant/qdrant   "./entrypoint.sh"   4 seconds ago   Up 4 seconds   0.0.0.0:6333-6334->6333-6334/tcp, :::6333-
6334->6333-6334/tcp   mystifying_beaver
moyankgiri@s3-workstation:~$
```

Figure 2: Docker container deployed for VectorDB

6.2 VectorDB User Interface

VectorDB is accessible via a User Interface at the same ports as shown in Figure 3

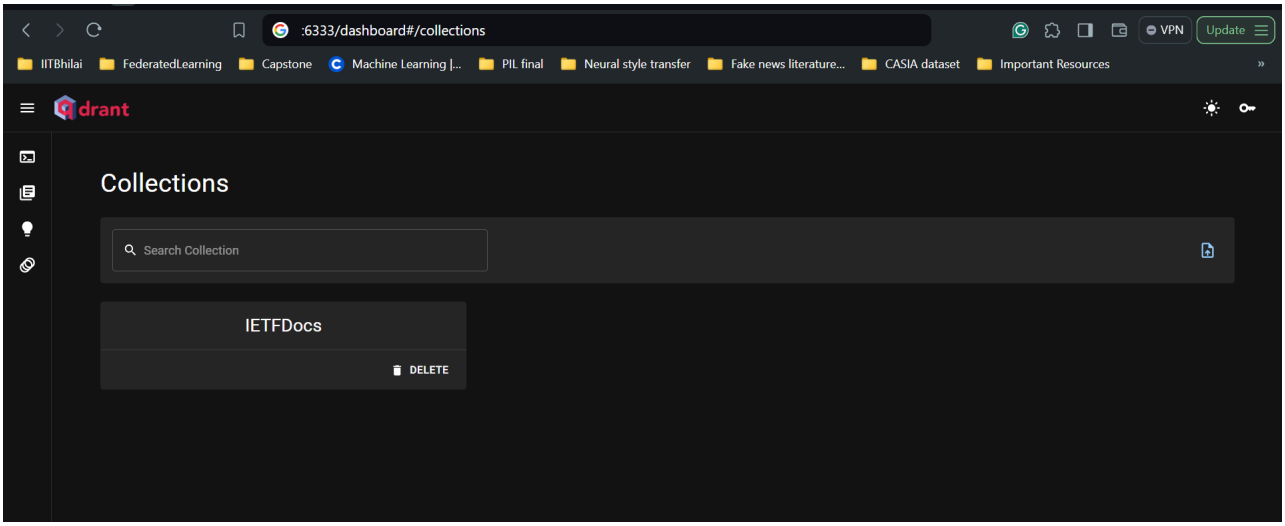

Figure 3: Qdrant UI

6.3 Vector storage on VectorDB

Vectors from IETF RFC 7884 are loaded using the embedding model as shown in Figure 4

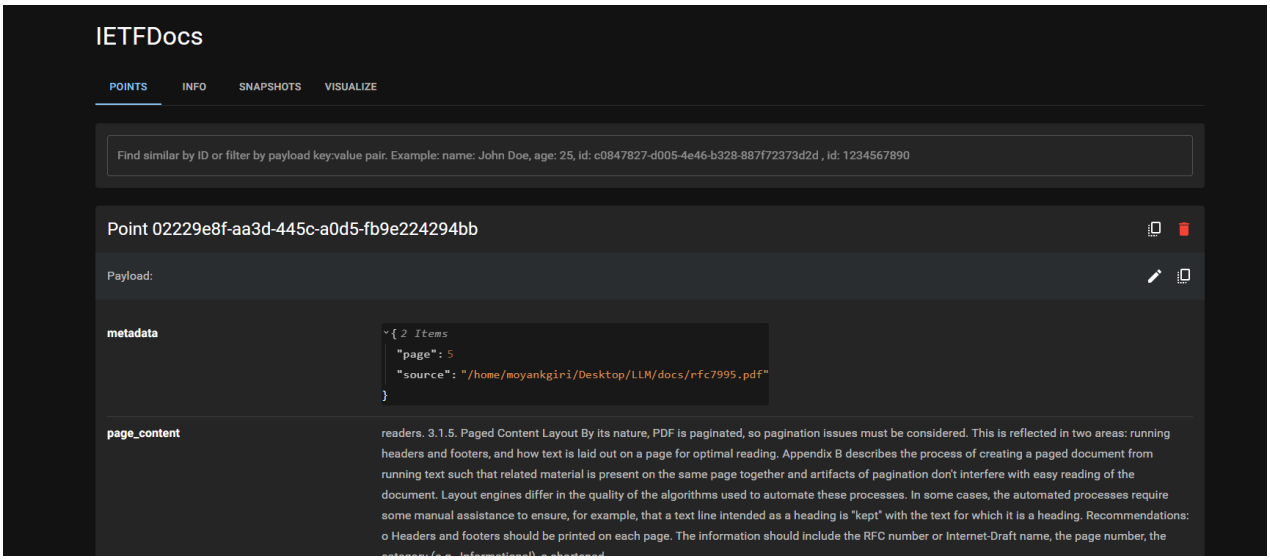

Figure 4: Sample Vectors stored in Qdrant

6.4 Vector Similarity search on VectorDB

Similarity search for vectors is also available allowing to generate context as shown in Figure 5

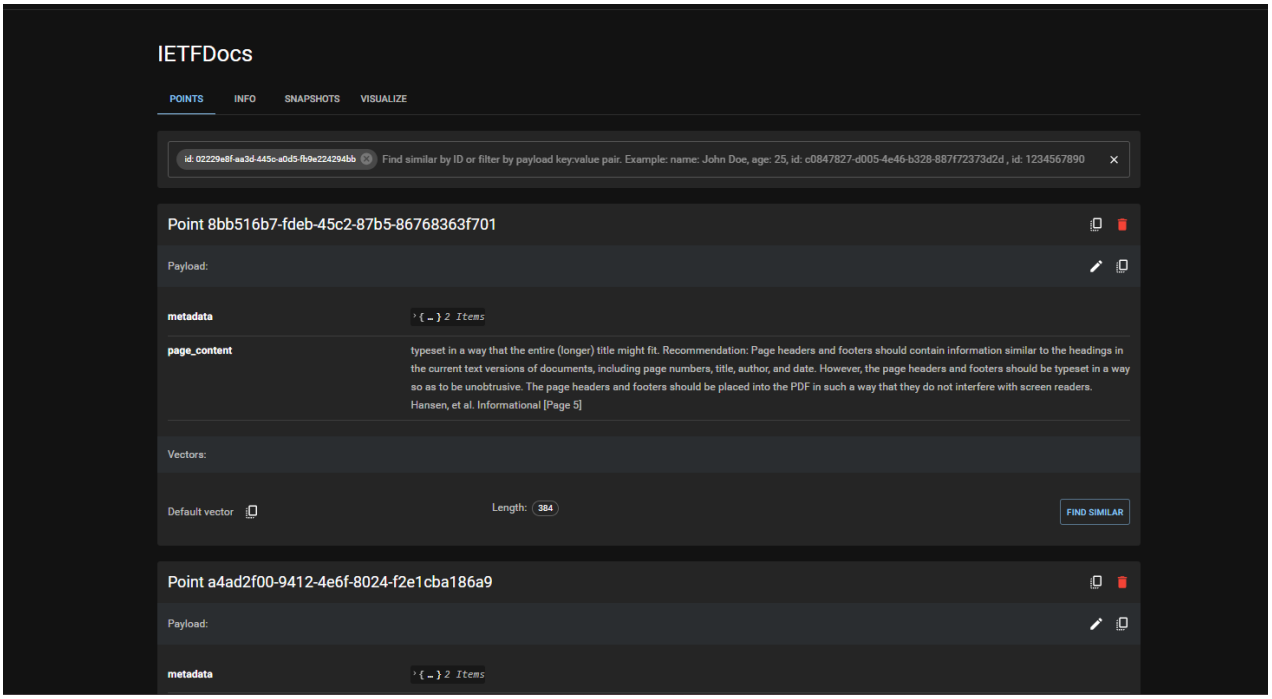

Figure 5: Similarity Search for Vector

6.5 LLM integration with VectorDB

Overall LLM's working with context from VectorDB for a user prompt is tested as shown in Figure 6

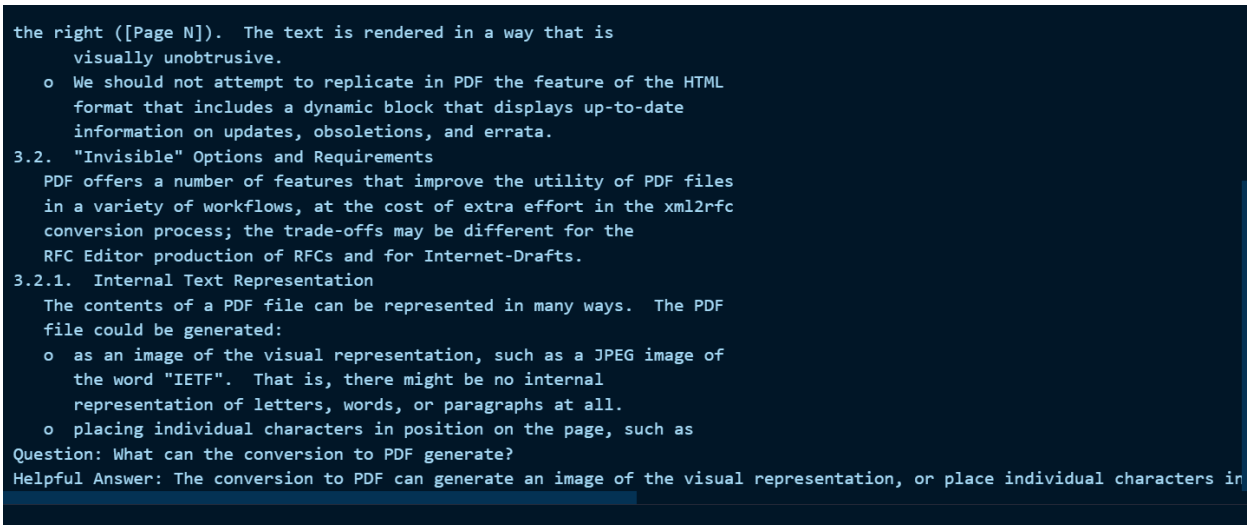

Figure 6: LLM Working with Qdrant context

6.6 TSNE Plots for Questions vs Context with Sample Query

This project also provides a comprehensive reasoning of usage of Questions instead of content for query matching. This is proven through TSNE plots shown in Figure 7 and 8 for 2 types of paragraphs

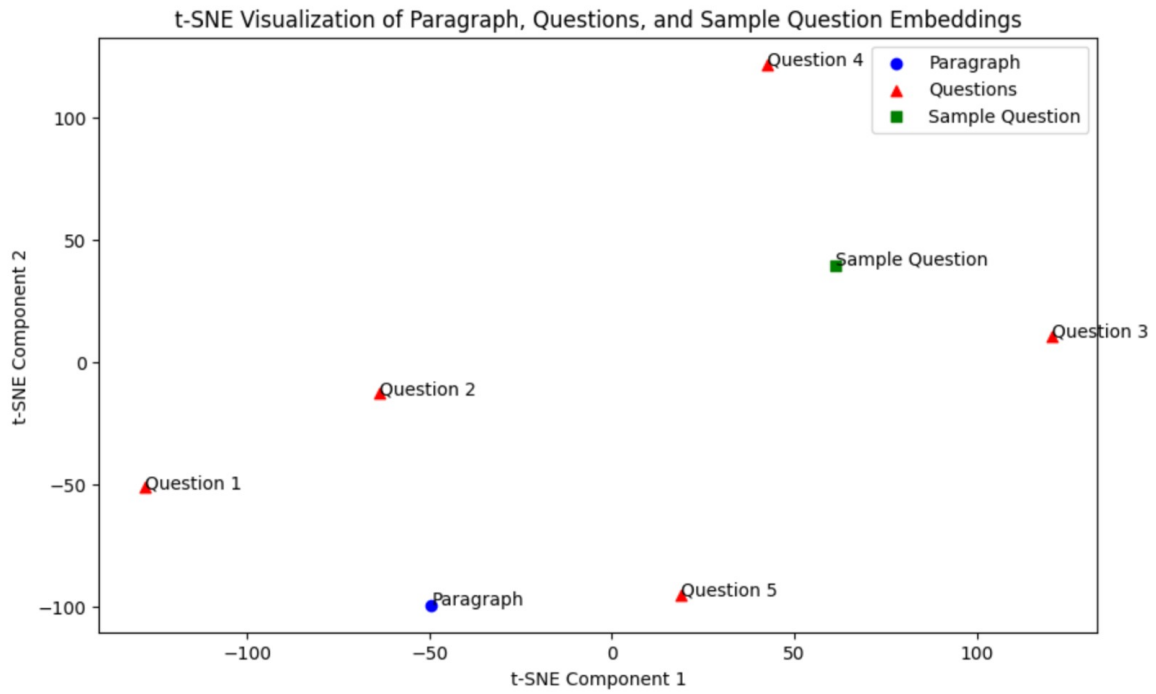

Figure 7: TSNE Plot1

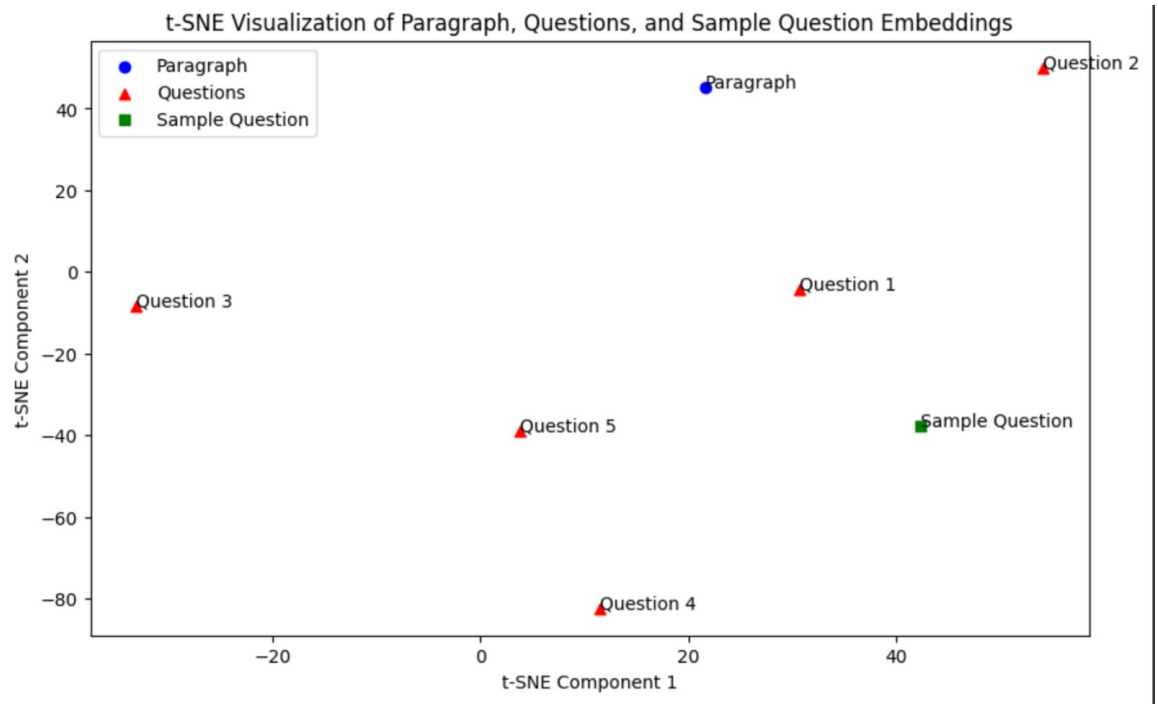

Figure 8: TSNE Plot2

6.7 User Interface

The implemented User Interface with user query, reference documents and history is shown in Figure 9

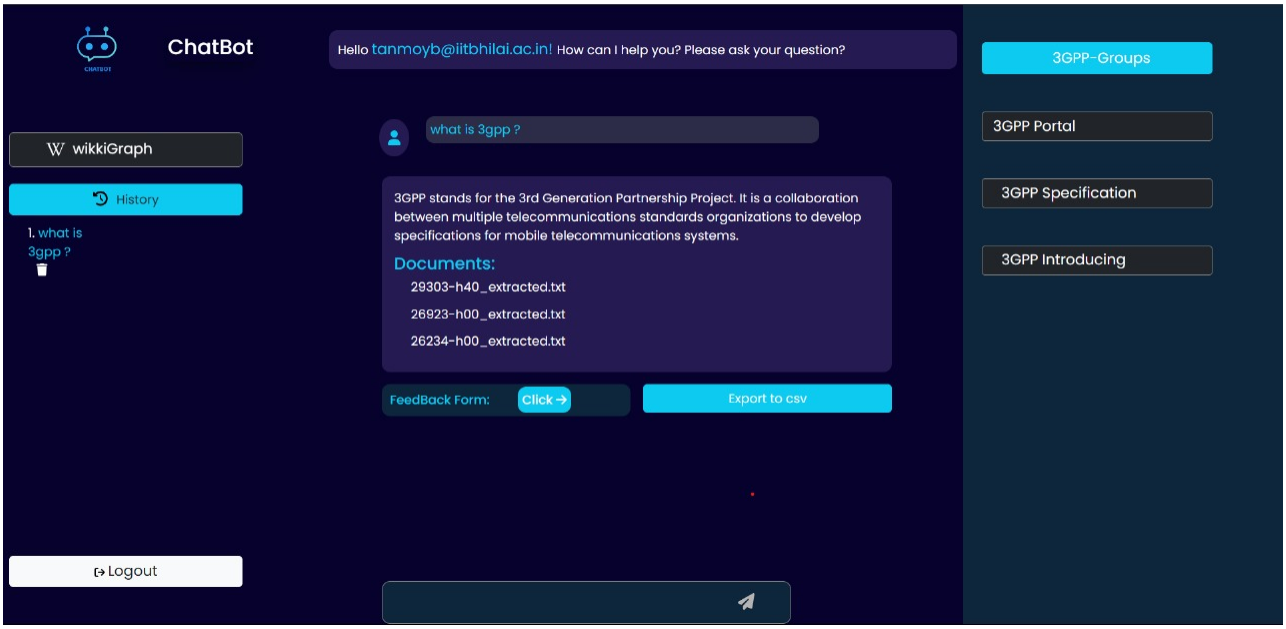

Figure 9: User Interface

7 Individual Contribution

The following table contains details of the proposed work distribution among the team members:

Contributor	Tasks
Moyank Giri	Generation of Vector Embeddings, Experimentation of Vector Embedding Models, Content-based and Question-based Vector DB Implementation, VectorDB and LLM Integration, Big Data Scripting, ETSI Documents collection
Harshit Kumar	Dataset creation from Public Forum (Reddit, Quora), TSNE Analysis of Question-Based Retrieval, Experimentation for chunking methods, Telecom Domain Dataset Retrieval, Ontology Research
Manish Rai	Dataset creation from 3GPP documents, Dataset creation from Public Forum (StackOverflow, StackExchange), Deep Analysis on relation between Chunk size and Question Quality, TSNE Analysis on various chunk size
Tanmoy Bhowmick	Deployment in Public Server, IETF Documents Download, Download Feature for History and Feedback, Creation of security certificates and auto-renewal feature, Bug Fix for display issues in variable screen resolutions

8 Conclusion

In developing our network troubleshooting chatbot, we have employed a sophisticated approach by combining the power of a Language Model (LLM) with a Retrieval-Augmented Generation (RAG) framework. This innovative fusion enables our chatbot to understand and generate contextually relevant responses and harness the strength of retrieval models for improved information retrieval.

This project showcases the potential of advanced language models and highlights the importance of incorporating retrieval mechanisms for real-world problem-solving. We hope that this chatbot development paves the way for more intelligent, context-aware, and effective solutions in the field of network troubleshooting

9 Git Repository

The git repository is available here: https://github.com/MoyankGiri/ComputerSystemDesignFinalProject

10 References

[1] arXiv:2401.08565

[2] Medium: Fine-tuning LLMs with Retrieval-Augmented Generation (RAG)

[3] Medium: RAG vs VectorDB

11 Appendix

11.1 Project Milestones and Timeline

Figure 10: Project Timeline