# A Retrieval-Augmented Generation (RAG) Chatbot personalized for VIT Vellore Operations

Rashi Solanki
School of Computer Science and Engineering
Vellore Institute of Technology
Tamil Nadu, India
rashisolanki2002@gmail.com

Veer Sanghavi
School of Computer Science and Engineering
Vellore Institute of Technology
Tamil Nadu, India
veersanghavi8@gmail.com

***Abstract*—This project presents a Retrieval Augmented Generation (RAG) ChatBot designed for university-related queries, answering questions about programs, events, admissions, and more. The ChatBot uses FAISS for similarity search, the Langchain library for context management, and the OpenAI GPT-3.5-turbo-0125 API for response generation. The system starts with data collection, compiling a broad dataset containing university information. OpenAI embeddings are used to transform text into vectors, which are indexed with FAISS for efficient similarity search. When a user submits a question, FAISS identifies the most relevant information from the vector database. The Langchain library links this context to the query, and the OpenAI GPT-3.5-turbo-0125 model generates natural responses. This approach allows the ChatBot to answer a variety of university-related questions accurately and efficiently. It also helps reduce administrative workload by automating responses to common inquiries. Future improvements might focus on expanding the dataset and refining response accuracy. Overall, this RAG ChatBot enhances communication and accessibility within the university environment.**

***Index Terms*—RAG, Langchain, FAISS, OpenAI**

## I. Introduction

The primary objective of implementing the Retrieval-Augmented Generation (RAG) framework at Vellore Institute of Technology (VIT) is to develop an advanced natural language processing (NLP) system tailored specifically to address topics related to the university. The rapid digital transformation in education has necessitated the adoption of innovative technologies to enhance student services and academic success. By integrating the OpenAI LLM model, our RAG system aims to provide comprehensive and accurate responses to inquiries, discussions, and queries pertaining to VIT. Unlike traditional question-answering systems, RAG combines a generative language model with a comprehensive knowledge base, enabling nuanced responses based on a wide array of information sources. This initiative seeks to empower students with a sophisticated conversational AI platform capable of addressing a wide range of topics related to academic programs, campus facilities, events, and administrative services at VIT. Through this scalable platform, we aspire to enhance accessibility, streamline communication channels, and foster engagement within the VIT community, thereby contributing to a more informed and connected university ecosystem. By implementing an RAG system, we aim to improve information retrieval efficiency, reduce administrative workload, and enhance the user experience for students and stakeholders, adapting to new information as it becomes available and evolving with the university's changing needs.

## II. Motivation

The motivation behind creating the RAG (Retrieval Augmented Generation) ChatBot for our university stemmed from the need to improve communication and information accessibility. With a complex and diverse set of university-related topics, ranging from academic programs to campus events, providing accurate information to students, faculty, and visitors had become a significant challenge. Traditional methods of answering queries through manual processes or static FAQs were no longer sufficient to meet the growing demands for immediate and precise responses. Recognizing these challenges, we sought a solution that could efficiently manage large volumes of information while providinguser-friendly experience. The RAG ChatBot emerged as the ideal approach, combining advanced technologies like FAISS for similarity search, the Langchain library for context management, and OpenAI GPT-3.5-turbo-0125 for generating natural language responses. This system offers a scalable and adaptable platform, capable of addressing the university's information needs while reducing the workload on administrative staff. By creating this ChatBot, we aim to foster a more connected and accessible university community, enhancing the overall experience for everyone involved.

## III. Literature Survey

The landscape of natural language processing (NLP) is continuously evolving, driven by innovative approaches aimed at enhancing the performance of text-based tasks. One such paradigm gaining prominence is Retrieval-Augmented Generation (RAG), as explored in the paper by Patrick Lewis et al. [1] This approach combines pre-trained parametric and non parametric memory for language generation, offering a nuanced solution for knowledge intensive NLP tasks. By integrating a pre-trained seq2seq model as parametric memory and a dense vector index of Wikipedia as non-parametric memory, RAG models demonstrate remarkable efficacy in open domain question

answering tasks. Additionally, they exhibit the ability to generate more specific, diverse, and factual language compared to traditional parametric-only seq2seq models, showcasing the potential of this synergistic approach. In a complementary study, Huayang Li et al. [2] present a survey on retrieval-augmented text generation, providing an overarching view of this emerging paradigm in computational linguistics. The paper outlines the generic framework of retrieval-augmented generation and reviews notable approaches across various NLP tasks, including dialogue response generation, machine translation, and other generation tasks. By synthesizing insights from diverse applications, the survey identifies promising directions for future research, highlighting opportunities to advance the capabilities of retrieval-augmented text generation further. Furthermore, Wayne Xin Zhao et al. [3] offer a comprehensive survey of Large Language Models (LLMs), focusing on the recent advancements in pre-trained models with significant parameter scales. This survey delves into various aspects of LLMs, including pre-training methodologies, adaptation tuning techniques, utilization strategies, and capacity evaluation methods. By examining the impact of LLMs on the AI community and outlining remaining challenges, the paper provides a roadmap for future research in this rapidly evolving field. The insights gleaned from these studies collectively contribute to a deeper understanding of retrieval augmented generation, its applications in knowledge-intensive NLP tasks, and the broader landscape of Large Language Models, offering valuable guidance for researchers and practitioners alike, as discussed by Zhao et al.

In their research, Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi [4] tackle the limitations of traditional generative dialogue models, which solely rely on input queries for response generation. Acknowledging that a single query can prompt multiple valid responses, the authors propose a novel framework that integrates information retrieval techniques to augment generative models. By retrieving similar dialogues from the training data, the model gains access to additional knowledge. However, the sheer volume of retrieved information can overwhelm generative models, leading to subpar performance. To overcome this challenge, the authors introduce a skeleton-to-response paradigm. Initially, a skeleton is extracted from the retrieved dialogues. Then, both the generated skeleton and the original query are utilized by a novel response generator to produce the final response. Experimental results demonstrate that this approach significantly enhances the informativeness of generated responses, highlighting its effectiveness in dialogue generation tasks. Authored by Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu [5], the proposed open-domain chatbot engine integrates Information Retrieval (IR) and Sequence to Sequence (Seq2Seq) based generation models. By employing an attentive Seq2Seq based rerank model to optimize the joint results, the system outperforms both IR and generation-based models in extensive experiments. The authors emphasize the significance of chatbots in various industries, particularly in e-commerce, where efficient customer question answering is crucial. They highlight the necessity of building an open-domain chatbot engine to enhance user experience, given the limitations of existing techniques such as IR and generation models in handling diverse and long-tail questions effectively.

## IV. Data Description

For the Retrieval-Augmented Generation (RAG) project at Vellore Institute of Technology (VIT), we have gathered a comprehensive dataset from a variety of sources to ensure the system's ability to provide detailed and accurate responses across multiple facets of university life. The data encompasses the following areas:

- **International Relations:** Information on exchange programs, collaborations with foreign universities, international student admissions, and global initiatives undertaken by VIT.
- **Academics:** Course details, curriculum structures, department-specific information, faculty qualifications, and academic policies.
- **Admissions:** Admission criteria, application procedures, scholarship opportunities, and frequently asked questions by prospective students.
- **Career Development Cell (CDC):** Details about placement statistics, internships, career counseling services, recruitment processes, and industry connections.
- **Hostels:** Accommodation facilities, hostel rules, room types, amenities provided, and information on dining services.
- **Clubs and Chapters/Events:** A comprehensive list of student clubs, professional chapters, cultural events, technical fests, and other extracurricular activities.
- **Faculty Information:** Profiles of faculty members, their research interests, publications, and contact details.
- **Campus Life:** General campus facilities, student services, health and wellness resources, campus safety, and student support systems.
- **VIT Research:** Research centers and opportunities for student involvement in research.
- **Examinations Details:** Grading systems, exam policies, and guidelines for students.
- **Library Services:** Information on library resources, digital access, study spaces, and research assistance.
- **Sports and Recreation:** Facilities for sports, fitness programs, and recreational activities.

## V. Methodology

### A. Data Collection and Cleaning

In our data collection process for the Retrieval-Augmented Generation (RAG) project at Vellore Institute of Technology (VIT), we adhered to rigorous standards to ensure the integrity and reliability of our dataset. Utilizing web scraping techniques like Beautiful Soup, we selectively gathered data from official VIT websites, ensuring that only openly available and non-confidential information was included. This approach

guaranteed the authenticity and reliability of the collected data, as it originated from trusted sources accessible to the public.

In cases where data was presented in non-readable PDF format, we employed Optical Character Recognition (OCR) techniques to extract text content, thereby making it accessible for analysis. By using web scraping and OCR, we ensured that our dataset encompassed a comprehensive range of openly available and authentic information from official VIT sources and other reliable sources, facilitating robust analysis and insights generation for our project.

### B. Comparison of LLMs

We compared different large language models (LLMs) based on accessibility, speed, model capacity, quality, and context length. OpenAI's cloud-based API enables seamless access and rapid inference, providing a significant advantage over locally deployed models. In contrast, locally installed models like Llama and Gemini LLM require substantial setup and maintenance, and they tend to have slower inference speeds. OpenAI's GPT models, with their larger number of parameters, generally produce more coherent and relevant text with fewer instances of hallucinations due to their vast training datasets. On the other hand, Llama and Gemini LLM models, with smaller parameter sizes, may not match the reliability and coherence of OpenAI's offerings, generating less accurate outputs.

Regarding context length, OpenAI's GPT-3.5-turbo-0125 supports a context length of up to 16,385 tokens, allowing it to maintain context over longer conversations or documents. In comparison, Llama 2 has a context length of 4,096 tokens, which may limit its ability to handle extensive interactions compared to OpenAI's models.

### C. Embedding Generation

We utilized OpenAI embeddings to transform the textual data into high-dimensional vectors. These embeddings facilitate efficient similarity searches and enhance the performance of the retrieval-augmented generation system.

### D. Prompt Engineering using LangChain

LangChain provides a collection of prompt templates designed to optimize interactions with large language models (LLMs). These templates offer structured instructions, significantly enhancing model performance. We customized these prompt templates to align with the specific needs of our VIT RAG system, ensuring that the generated responses are relevant and contextually appropriate.

To enable fast and efficient similarity searches within the retrieved documents or datasets, we integrated FAISS (Facebook AI Similarity Search). The FAISS enhances the system's ability to fetch contextually relevant information quickly. By integrating LangChain, we leveraged its advanced language generation capabilities, allowing the model to produce coherent and contextually accurate text based on the retrieved information.

### E. Combined Architecture

We developed a combined architecture that integrates LangChain's generation module with FAISS-powered retrieval. This hybrid approach enables the OpenAI model to fetch and utilize external knowledge efficiently, ensuring more accurate and contextually relevant responses.

### F. Building the UI

Using Streamlit, we constructed an intuitive user interface (UI). Streamlit's Pythonic syntax allowed us to build the UI quickly using familiar Python code, ensuring ease of development and maintenance. Additionally, Streamlit automatically manages the reactivity of UI components, eliminating the need for manual event handling or state management, simplifying the development process, and ensuring a smooth user experience.

## VI. High Level Solution Flow

### A. Embedding Generation

Creating embeddings involves representing words or tokens in a high-dimensional vector space where semantically similar words are closer together. OpenAI's approach to creating embeddings involves tokenization, learning embeddings through a trainable process, incorporating positional embeddings, and using self-attention mechanisms.

*1) Tokenization:*

- Tokenization involves breaking down the text into smaller units such as words or subwords.
- For example, in the sentence "The cat sat on the mat," tokenization might result in ['The', 'cat', 'sat', 'on', 'the', 'mat'].
- This step ensures that each word or subword is represented as a discrete unit.
- If necessary even words are broken down and separated into smaller tokens for better learning of the model.

*2) Learning Embeddings:*

- After tokenization, each token is assigned an initial embedding vector randomly.
- These initial embeddings are then updated during the training process of a neural network model.
- The model learns to adjust the embeddings such that they capture meaningful relationships between tokens based on the context in which they appear.
- For example, in the context of "cat" and "dog," the model learns that these words are often associated with each other in sentences describing a scene.

*3) Positional Embeddings:*

- Positional embeddings are added to the token embeddings to encode the position of each token in the sequence.
- This step is crucial because it allows the model to differentiate between tokens based on their position in the sequence, preserving sequential information.
- Positional embeddings are typically learned alongside token embeddings.
- For example, in a sentence of length N, each token would have an associated positional embedding vector indicating its position from 1 to N.

*4) Self-Attention Mechanism: Mathematical Explanation:* In self-attention, each word in the input sequence is associated with three vectors: the query vector, the key vector, and the value vector. These vectors are linear transformations of the input embeddings.

- **Dot Product**: To compute the attention scores, the query vector of a word is dot-producted with the key vectors of all other words in the sequence. This yields a score that represents the relevance of each word to the current word.
- **Softmax**: The attention scores are then passed through a softmax function to obtain attention weights, which sum up to 1. These weights determine how much each word contributes to the representation of the current word.
- **Weighted Sum**: Finally, the weighted sum of the value vectors, where the weights are determined by the attention weights, produces the context vector for the current word. This context vector captures the information from the entire input sequence relevant to the current word.

*5) Benefits of Self-attention:*

- Global Context: Self-attention allows each word in the sequence to consider the entire context of the input when generating its representation, enabling the model to capture long-range dependencies.
- Adaptability: The attention mechanism is adaptive, meaning it can assign different weights to different parts of the input sequence based on their relevance to the current word, making it effective for a wide range of NLP tasks.

### *B. Retrieval using FAISS*

FAISS, or Facebook AI Similarity Search, is a library developed by Facebook's AI Research (FAIR) team to efficiently search for vectors in large-scale datasets. It's particularly useful for tasks like similarity search, where you want to find items that are most similar to a given query vector. FAISS employs techniques like indexing and quantization to achieve fast and memory-efficient search.

*1) FAISS Workflow:*

1) IndexIVF (Inverted File Index):
   - FAISS typically starts by partitioning the vector space into regions using k-means clustering. Each region is represented by a centroid.
   - Mathematically, the k-means algorithm iteratively updates centroids to minimize the sum of squared distances between each vector and its assigned centroid.
   - This process creates a Voronoi tessellation of the vector space, where each centroid acts as a representative for its region.
   - These centroids facilitate efficient search by dividing the search space into smaller, manageable partitions.
2) Vector Assignment:
   - Once the regions are defined, vectors are assigned to the nearest centroid based on the Euclidean distance (L2 distance) between each vector and the centroids.
   - Given a set of centroids $C = \{c_1, c_2, ..., c_k\}$ and a query vector $q$, the vector assignment involves finding the centroid $c_i$ that minimizes the Euclidean distance $||q - c_i||$.
   - This assignment creates an inverted file index, where each cell holds the list of vectors assigned to a particular centroid.
   - By associating vectors with centroids, FAISS reduces the search space, making it more efficient to search for similar vectors.
3) Efficient Retrieval:
   - During retrieval, FAISS employs techniques to efficiently find the nearest neighbors of a query vector.
   - It calculates the Euclidean distance between the query vector and the centroids of different regions.
   - Based on these distances, FAISS selects regions to probe more thoroughly, focusing the search on potentially relevant areas of the dataset.
   - Within these regions, FAISS then retrieves the nearest neighbors using algorithms optimized for high-dimensional search.

*2) Advantages of FAISS:*

1) Fast Search:
   - FAISS enables rapid similarity search in high-dimensional spaces, making it suitable for tasks like image search, recommendation systems, and natural language processing.
2) Efficient Storage:
   - FAISS allows vectors to be stored in a quantized format, reducing memory usage while preserving search quality to a large extent.
3) Scalability:
   - FAISS can efficiently handle large datasets with millions of vectors, making it suitable for real-world applications with extensive data volumes.
4) Flexible Indexing:
   - FAISS supports various indexing methods, including flat (where all vectors are stored in a single index structure) and IndexIVF (where vectors are partitioned into regions).
5) Distance Metric Support:
   - FAISS provides support for various distance metrics, including L2 (Euclidean distance) and cosine similarity, allowing users to choose the most appropriate metric for their application.

*3) Retrieval:*

1) Query Processing: When a user submits a query, it's transformed into a high-dimensional vector using the same embedding techniques used for indexing.
2) Voronoi Region Selection: Using the representation of the query vector, the Voronoi cell containing the query's nearest centroid is identified. This focuses the search on potentially relevant documents closer to the query in the vector space.
3) Efficient Retrieval: Within the selected Voronoi region, retrieval algorithms like FAISS are employed to effi-

ciently find the nearest neighbors of the query vector. This narrows down the search space, improving efficiency in high-dimensional spaces.
4) Ranking and Generation: Retrieved documents or passages are ranked based on their similarity to the query and fed into a language generation model like GPT for response or document generation.

In summary, FAISS offers a powerful set of tools for similarity search, leveraging indexing, quantization, and efficient retrieval techniques to handle large-scale datasets effectively. Its flexibility, scalability, and support for various distance metrics make it a valuable tool for a wide range of applications in machine learning and data analysis. By integrating Voronoi tessellations into the retrieval process, RAG enhances the efficiency and effectiveness of finding relevant content based on user queries. This approach optimizes the retrieval step by focusing the search on specific regions of the vector space, leading to faster and more accurate retrieval results.

### C. Generation

1) **Input Query**: The user inputs a query related to Vellore Institute of Technology (VIT), such as admission process, courses offered, campus facilities, etc.
2) **Contextualization**: The Generation Component begins by contextualizing the user's query, ensuring understanding in isolation without relying on previous chat history. This involves using a system prompt and the OpenAI GPT-3.5-turbo-0125 model to frame the query appropriately.
3) **Retrieval**: The system retrieves relevant information from the knowledge base regarding the user's query. This information includes details about VIT, such as its history, programs, facilities, events, and any other pertinent data.
4) **Retrieval-Aware Answer Generation**: Using the retrieved context from the knowledge base and the provided prompt, the Generation Component generates an answer to the user's query. The OpenAI GPT-3.5-turbo-0125 model takes into account the retrieved context, chat history, prompt, and input question to craft a response tailored to the user's query.
5) **History-Aware Answer Generation**: To further enhance the response, the Generation Component incorporates the chat history. This enables the model to provide more accurate and relevant responses based on the conversation context, maintaining coherence throughout the interaction.
6) **16k Context Length Advantage**: The use of the OpenAI GPT-3.5-turbo-0125 model with a context length of 16k characters allows for the consideration of extensive information when generating responses. This ensures that the model can access a wide range of knowledge about Vellore Institute of Technology, resulting in comprehensive and detailed answers to user queries.
7) **Output Response**: Finally, the Generation Component outputs a response to the user's query, providing comprehensive and informative answers related to Vellore Institute of Technology. The response is generated with consideration of the retrieved context, chat history, and the advantages of a 16k context length, ensuring high-quality interactions.

## VII. Constraints, Alternatives, and Tradeoffs

### A. Constraints

- **Data Quality and Consistency:** The accuracy of the ChatBot's responses depends on the quality and consistency of the data used to build the vector database. Incomplete or outdated data can lead to incorrect or misleading answers.
- **Computation and Resource Requirements:** The use of FAISS, OpenAI embeddings, and GPT-3.5-turbo-0125 requires significant computational resources and memory. This can lead to increased costs, especially if the system needs to scale with demand.
- **Latency and Response Time:** The retrieval and generation process can introduce latency, impacting user experience. Response time must be minimized to ensure a seamless interaction.
- **Privacy and Security:** Handling sensitive university data requires stringent security measures. Compliance with data protection regulations (like GDPR) is essential to protect user information and maintain trust.

### B. Alternatives

- **Rule-Based ChatBots:** Instead of a RAG system, a rule-based ChatBot could be used. This approach relies on predefined scripts and logic but may lack flexibility and context-awareness.
- **Traditional Retrieval Systems:** A basic retrieval system that doesn't use advanced embeddings or similarity search might be simpler and less resource-intensive, though it may not be as accurate or contextually aware.
- **Different Language Models:** Instead of GPT-3.5-turbo-0125, other language models could be used, potentially reducing costs and resource requirements, but possibly sacrificing response quality.

### C. Tradeoffs

- **Accuracy vs. Cost:** Using advanced AI models and embeddings ensures high accuracy, but it also increases costs due to computational and resource requirements. Simplifying the model or using smaller embeddings could reduce costs but at the expense of accuracy.
- **Response Quality vs. Latency:** Higher-quality responses from complex models may lead to increased latency. Reducing response time might require sacrificing some level of response complexity or accuracy.
- **Flexibility vs. Complexity:** A flexible system like RAG allows for more dynamic responses but introduces complexity in system design and maintenance. A simpler system may be easier to manage but less adaptable to varying queries.

These constraints, alternatives, and tradeoffs must be carefully considered when developing and implementing the RAG ChatBot for a university setting.

## VIII. Results

## IX. Use Case Scenarios

### A. Easy Access to Faculty Contact Information

One of the significant advantages of the RAG ChatBot is its ability to provide quick access to faculty contact information. Traditionally, students would need to navigate through various sections of the university website to find email addresses or contact details of faculty members. This process can be time-consuming and frustrating. However, with the RAG ChatBot, students can simply ask for the email address of a specific faculty member or inquire about the contact details of professors in a particular department. The ChatBot efficiently retrieves the relevant information from its database and provides an accurate response in seconds. For instance, a student needing to contact Dr. Rajkumar R from the School of Computer Science and Engineering (SCOPE) can quickly obtain her email by asking the ChatBot, eliminating the need to search through multiple web pages.

### B. Enhanced Networking with Faculty Members

The RAG ChatBot also plays a crucial role in facilitating better networking opportunities between students and faculty members. Often, students may only interact with faculty who teach their courses, missing out on the broader expertise available within the university. By leveraging the ChatBot, students can ask specific questions such as, "Who should I approach if I have a problem in Natural Language Processing (NLP)?" The ChatBot will provide a list of faculty members specializing in NLP, including their research interests and contact details. This enables students to connect with professors whose expertise aligns with their academic and research interests, fostering a more enriched educational experience. Consequently, students can gain more knowledge and exposure by networking with a diverse group of faculty members, beyond those who directly teach their classes.

### C. Discovering Clubs and Chapters Based on Interests

With over 180 clubs and chapters at VIT Vellore, finding the right one can be overwhelming for students. The RAG ChatBot simplifies this process by helping students discover clubs and chapters that align with their interests. For example, a student interested in aviation can ask, "I am interested in aviation, which club or chapter should I join?" The ChatBot will provide a list of relevant clubs or chapters, along with brief descriptions. This feature not only saves time but also ensures that students find groups that match their passions and interests, enhancing their campus experience. By connecting students with the right clubs and chapters, the ChatBot promotes a more engaged and active student community, helping students to develop skills, pursue hobbies, and network with like-minded peers.

## X. Cost Analysis

The cost analysis for this Retrieval Augmented Generation (RAG) project is centered around the expenditure associated with the use of OpenAI's GPT-3.5-turbo-0125 model. The cost structure for this model involves charges for both input and output tokens. Below, we break down the costs based on the project's specific requirements.

- **Input Tokens:** Given that the cost for input tokens is $0.50 per 1 million tokens, the expenditure for this component is straightforward:
  - **Cost of Input Tokens:** We have used 1 million tokens, so the cost for input tokens is calculated as $0.50 per 1 million tokens. Thus, our expenditure was only $0.50. We spent more than just $0.5 in our testing phase, which is an approximate of $1.0.
- **Output Tokens:** Output tokens refer to the text generated by the model in response to your inputs. As with the input tokens, the pricing for output tokens is based on the volume of tokens used. The cost for output tokens in this model is $1.50 per 1 million tokens.
  - **Cost of Output Tokens:** The output tokens' cost depends on the volume generated during project execution. This is not defined because it wholly depends on the questions asked and how much the model is asked to generate. During this project, we spent an approximate of $4.5 dollars, that is, we generated around 3 million tokens to make sure our testing is done right.

**Total Cost:** Only $5.5 spent in creating this whole project!