

# PatriotPilot : Your Virtual GMU Guide

<b>Rohit Ganesh</b> G01451823 MS CS George Mason University <a href="mailto:rghanesh5@gmu.edu">rghanesh5@gmu.edu</a>	<b>Nitesh Thota</b> G01424648 MS CS George Mason University <a href="mailto:nthota2@gmu.edu">nthota2@gmu.edu</a>	<b>Santhosh Sumathi Elumalai</b> G01446000 MS CS George Mason University <a href="mailto:ssumathi@gmu.edu">ssumathi@gmu.edu</a>
--	--	---

## Repository Link

Find the GitHub repository for this project at the following link:  
<https://github.com/santhoshse99/PatriotPilot>

## 1 Introduction

### 1.1 Task / Research Question Description

We aim to create a chatbot that assists users in navigating George Mason University's vast online resources, providing personalized recommendations on topics like courses, faculty information, clubs, and department information. The goal is to reduce the complexity of finding relevant information by delivering direct, efficient and personalized responses.

### 1.2 Motivation and Limitations of existing work

University websites can be overwhelming due to their scale and complexity, often leading to frustrating user experiences. Current systems like FAQs or static chatbots fail to offer personalization or efficient navigation, requiring users to manually sift through multiple pages. Our approach differs by focusing on an intelligent chatbot capable of retrieving relevant information and personalizing the user experience. Previous approaches also lacked the dynamic adaptability needed for real-time queries, often relying on static responses.

### 1.3 Proposed Approach

Our approach integrates NLP with Retrieval-Augmented Generation (RAG), leveraging the E5-large-v2 embedding model to encode both user queries and preprocessed university resource data into dense vectors. These embeddings are indexed using FAISS, enabling fast and accurate retrieval of semantically relevant information. The chatbot utilizes Qwen-2.5-14B, a fine-tuned large lan-

guage model, to generate contextually appropriate responses based on the retrieved data. The retrieval component, enhanced with semantic relevance filtering, ensures the accuracy and relevance of the retrieved content, while the generative model synthesizes concise, professional, and user-friendly answers, bridging the gap between structured data and conversational interaction.

### 1.4 Likely challenges and mitigations

- Challenge:** Contextual Irrelevance in Retrieval  
**Mitigation:** Enhanced FAISS retrieval by prioritizing query-specific keywords and dynamically filtering less relevant results, ensuring more precise contextual matches.
- Challenge:** Limited Context Length in Generation  
**Mitigation:** Trimmed and prioritized retrieved contexts dynamically to fit within model token limits, ensuring key information was preserved for accurate response generation.
- Challenge:** Dataset Quality Issues  
**Mitigation:** Improved preprocessing pipelines to standardize text formats and remove noisy or redundant data, enhancing retrieval and generation accuracy.
- Challenge:** Integration and Deployment Errors  
**Mitigation:** Adopted a robust Flask-based backend with efficient error handling and API integration, enabling seamless communication between the frontend and backend.

## 2 Related Work

- "Chatbots Development Using Natural Language Processing: A Review" This paper provides

a comprehensive review of various chatbot development techniques utilizing NLP. It highlights how chatbots have evolved, discussing common NLP techniques such as tokenization, named entity recognition, and sentiment analysis. The paper also explores the use of machine learning models to improve chatbot capabilities for customer support and human-machine interactions **Abdulla et al. (2023)** Chatbots Development Using Natural Language Processing: A Review . For more details, see the full paper at <https://ieeexplore.ieee.org/document/10017592>.

Difference: Our project differentiates itself by focusing on a university-specific chatbot with a recommendation system and retrieval-augmented generation (RAG), which is not explored in this review.

2. "A Literature Survey of Recent Advances in Chatbots" This paper surveys the state-of-the-art advancements in chatbot development, particularly focusing on deep learning models like GPT, BERT, and various Transformer-based architectures. It discusses the application of these models in a wide array of domains, from customer service to healthcare

**Guendalina Caldarini (2023)** discuss recent advancements in chatbots.

. For more details, see the full paper at <https://www.mdpi.com/2078-2489/13/1/41>.

Difference: While this survey focuses on general-purpose chatbot advancements, our project tailors its chatbot to the specific needs of students and faculty at George Mason University, integrating both NLP and a recommendation system for personalized responses.

3. "A Complete Survey on LLM-based AI Chatbots". This paper provides an in-depth overview of chatbots powered by Large Language Models (LLMs), covering their evolution and applications across education, research, and healthcare domains. It also highlights the technical challenges and ethical concerns associated with LLM-based chatbots. **Dam et al.**

(2024) A Complete Survey on LLM-based AI Chatbots. For more details, see the full paper at <https://arxiv.org/abs/2406.16937>.

Difference: While this paper covers LLM-based chatbots across different sectors, our project specifically focuses on improving information retrieval for university-related queries using RAG.

4. "RAG: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". This paper focuses on enhancing the capabilities of large language models by combining them with a retrieval system like FAISS. It demonstrates how LLMs can retrieve relevant external knowledge, integrating it into the generation process to improve performance on knowledge-intensive tasks. **Lewis et al. (2021)** provide a detailed exploration of this approach. For more details, see the full paper at <https://arxiv.org/abs/2005.11401>.

Difference: Our project applies similar retrieval techniques within the university context, utilizing Qwen-2.5-14-Instruct for personalized responses.

### 3 Methodology

#### Data Collection and Preprocessing

We collected data by manually extracting various university resources from George Mason University's website. The collected data, including academic and administrative resources, is stored in structured JSON files. For cases where data was complex or inaccessible, we manually curated and formatted it into JSON. These JSON files retain a hierarchical structure to preserve context, such as headings and sections that provide important contextual information.

#### Preprocessing

The preprocessing pipeline was designed to prepare raw university data for efficient embedding and retrieval while preserving essential context. The steps included:

- **Tokenization:** Text data was tokenized into smaller units using NLTK to facilitate subsequent preprocessing steps.
- **Lowercasing:** All text was converted to lowercase to ensure uniformity and prevent case-

sensitive mismatches during embedding and retrieval.

- **Preserving Critical Information:** Certain elements, such as emails, URLs, symbols ('@', '/', '.'), and numerical data, were retained during preprocessing to avoid loss of essential information.
- **Redundant Data Removal:** Redundant or noisy data was removed to enhance the quality of both retrieval and response generation. Manual checks were applied to ensure that important contextual information remained intact.
- **Contextual Labeling:** Key data elements were labeled with descriptive tags, and parent keys were assigned meaningful names. This ensured that hierarchical and relational structures within the data were preserved, facilitating the embedding and retrieval process.
- **Chunking:** Processed text was split into manageable chunks of up to 512 tokens to maintain the integrity of information and adhere to the input limits of the embedding and language models.
- **Error Handling:** Inconsistent data formats were identified and corrected, ensuring uniformity across the dataset. This included addressing incomplete entries and merging fragmented data.
- **Optimization for Retrieval:** Stopwords were selectively removed except in contexts where they added semantic value, such as in phrases or identifiers.

This robust preprocessing ensured that the text data fed into the system retained its semantic richness, thereby improving the quality of retrieval and the accuracy of generated responses.

### Embedding and Indexing

We employed the *e5-large-v2* embedding model to convert preprocessed text chunks and user queries into dense 1024-dimensional vectors, capturing their semantic meaning. The precomputed embeddings were indexed using FAISS, enabling efficient similarity searches to retrieve relevant text chunks. At runtime, user queries were embedded and matched against this index,

retrieving the top- $k$  relevant results based on cosine similarity. To enhance relevance, additional filtering prioritized query-specific keywords. This approach ensured accurate and efficient retrieval, critical for real-time response generation in our chatbot system.

### Embedding Model

We utilized the *e5-large-v2* embedding model from SentenceTransformers due to its exceptional ability to generate high-quality sentence-level embeddings. The model effectively captured semantic relationships in text, enabling robust representation of both user queries and dataset chunks. Its state-of-the-art performance ensured accurate retrieval of contextually relevant information, forming the foundation for precise and meaningful responses.

### FAISS Index

We used FAISS (Facebook AI Similarity Search) to index the embedded chunks for efficient similarity search. FAISS allows the system to quickly retrieve relevant text chunks when a user query is embedded and compared with the stored vectors. The embeddings of the data are pre-computed and stored, while the user queries are embedded at runtime and matched against this index. This method allows real-time query-response interaction without re-embedding the full dataset.

### Retrieval-Augmented Generation (RAG)

To provide relevant and accurate responses to user queries, we employed a Retrieval-Augmented Generation (RAG) approach:

#### Context Retrieval

When a user query is submitted, it is embedded using the E5-large-v2 model, and the resulting vector is matched against the pre-embedded data chunks in the FAISS index. The top- $k$  most relevant chunks are retrieved based on cosine similarity.

#### Language Model

We employed the Qwen-14B-instruct language model to generate responses, leveraging its advanced natural language understanding and generation capabilities. This model was fine-tuned for our specific use case to ensure that it could interpret user queries accurately and generate contextually appropriate answers based on the

retrieved information. Its robust architecture allowed for precise and coherent responses, enhancing overall user satisfaction.

### Fine-Tuning with LoRA

To adapt the Qwen-14B-instruct model to our specific task, we initially employed Low-Rank Adaptation (LoRA), a lightweight fine-tuning technique. This approach allowed us to update only a subset of the model’s parameters, significantly reducing computational overhead while maintaining performance. Using a task-specific dataset, the model was fine-tuned for 5 epochs with a small learning rate of  $1 \times 10^{-5}$ . However, due to the limited amount of training data, the fine-tuned model performed worse than the out-of-the-box Qwen-14B-instruct model. Consequently, we decided to proceed with the pre-trained model instead.

We hypothesize that with a larger dataset and full fine-tuning, more noticeable improvements could be achieved, potentially making this approach more viable for similar tasks in the future. .

### Model Inference

For each user query, the system constructs a dynamic prompt by combining the retrieved context from the FAISS index with the user’s input. This prompt is then processed by the fine-tuned Qwen-14B-instruct model to generate a concise and contextually relevant response. The inference pipeline ensures that responses are tailored to the query while leveraging retrieved information to maintain factual accuracy and coherence.

### Prompt Design

The prompt is designed to provide the context first, followed by the user’s question. This structure ensures that the model has all the necessary information to generate a meaningful response. For example:

**Context: [This is the context retrieved from FAISS Index]**

```
contactinfo      administrationgraduateadvising    location    engineering  
building room 4300 contactinfo administrationgraduateadvising mapurl  
https://csgmuedu/about/visit/the/department  
contactinfo      undergraduateadvising  
location buchanan hall room d215 contactinfo undergraduateadvising mapurl
```

```
https://csgmuedu/about/visit/the/department  
contactinfo phononenumber 703 9931530  
contactinfo mailingaddress department department computer science  
contactinfo mailingaddress university george mason university contactinfo  
mailingaddress address
```

**User Query:** What is the computer science department phone number?

**Response:** Answer: 703 9931530

### Generation Configuration

The following parameters were utilized during model inference to ensure that the generated responses were coherent, relevant, and concise:

- **Max Length:** 150 tokens (ensures responses remain focused while capturing sufficient detail).
- **Num Return Sequences:** 1 (generates a single response per query to maintain clarity).
- **Temperature:** 0.5 (optimizes response coherence by reducing randomness in token selection).
- **Top-k:** 50 (restricts the model’s token choices to the top 50 with the highest probabilities for meaningful generation).
- **Pad Token ID:** Utilizes the Qwen tokenizer’s `eos_token_id` to ensure no truncation errors in generation.

These configurations were fine-tuned to balance accuracy, fluency, and efficiency in the system’s responses.

### 3.1 Evaluation

#### Evaluation Results and Analysis

##### Results Overview

**Average Precision: 0.3**  
**Average Recall: 0.6**  
**Average F1-Score: 0.4**

Figure 1: Overall Evaluation Metrics.

### **Retrieval Evaluation:**

- **Precision:** 0.30
- **Recall:** 0.60
- **F1-Score:** 0.40

### **Text Generation Evaluation:**

- **BLEU Score:** 0.06
- **ROUGE-1 F-Measure:** 0.39
- **ROUGE-2 F-Measure:** 0.31
- **ROUGE-L F-Measure:** 0.37

### **Interpretation of Results**

**Retrieval Evaluation:** The retrieval module demonstrated moderate recall (0.60), indicating its capability to retrieve relevant contexts effectively. However, the low precision (0.30) highlights that the retrieved contexts often included irrelevant or noisy information. This imbalance between precision and recall resulted in a low F1-score (0.40), reflecting challenges in filtering out irrelevant content during the retrieval process.

**Text Generation Evaluation:** The BLEU score (0.06) indicates that the generated responses deviated significantly from the expected answers, suggesting issues with accurately replicating the structure and vocabulary of reference responses. The ROUGE scores (**ROUGE-1: 0.39**, **ROUGE-2: 0.31**, **ROUGE-L: 0.37**) suggest that while the generated responses captured some relevant information, they lacked consistency and completeness compared to reference answers.

### **Human Evaluation**

We conducted a qualitative evaluation by asking our peers to interact with the chatbot and provide feedback. Most evaluators noted that the chatbot was capable of answering simple and straightforward queries accurately. However, for complex or nuanced queries, the chatbot occasionally hallucinated and provided incorrect responses. This feedback highlights areas for improvement, particularly in reducing response hallucination and enhancing consistency.

### **Reasons for Low Performance**

- **Embedding Model Limitations:** The embedding model (*E5-large-v2*) struggled with task-specific context extraction, leading to retrieval of irrelevant or fragmented information.
- **Noisy Preprocessing:** The preprocessing introduced noise through aggressive flattening and labeling, diluting meaningful information.
- **Chunk Overlap and Context Loss:** The static chunking strategy caused information loss, impacting retrieval and leading to incomplete responses.
- **Prompt Engineering Challenges:** The language model (*Qwen-2.5-14B-Instruct*) required better fine-tuning for domain-specific queries, contributing to deviations from expected responses.
- **Dataset Quality:** Inconsistencies and stripped context in the dataset affected model training and testing alignment, reducing system performance.

### **Room for Improvement**

- **Improved Embedding Model:** Explore domain-specific or more robust embeddings with better context extraction capabilities.
- **Enhanced Preprocessing:** Reduce noise and preserve semantic structure through refined preprocessing techniques.
- **Chunking Strategy:** Implement overlapping or dynamic chunking to mitigate context loss during retrieval.
- **Fine-Tuning the LLM:** Optimize the Qwen model with domain-specific data and prompts for concise and accurate responses.
- **Feedback Loop:** Introduce a mechanism to iteratively improve the retrieval and generation components based on user interactions.
- **Expanded Evaluation Metrics:** Include additional evaluation metrics such as METEOR and semantic similarity for more comprehensive assessments.
- **Dataset Refinement:** Curate high-quality datasets with consistent context alignment for training and evaluation.

## Conclusion of Evaluation

The results underscore the challenges faced in both retrieval and generation tasks, primarily due to noisy preprocessing, embedding limitations, and prompt engineering shortcomings. While the current performance demonstrates room for improvement, the outlined strategies provide a roadmap for enhancing the chatbot’s accuracy, reliability, and usability.

## 4 Experiments

### 4.1 Datasets

All the data used in this project is publicly available on George Mason University’s website. For the first phase, we are focusing on data from the Computer Science Department’s website, which includes information on faculty, advising, courses, and other general resources. While the data is public, we manually curated it to ensure it is well-structured and formatted for effective retrieval using FAISS. The manual structuring was necessary to provide clean and context-rich data for embeddings.

#### Availability

The dataset is not directly available as a single resource since it was scraped from different sections of the website and manually structured into JSON files.

#### Train / Test Splits

We manually curated question-answer pairs derived from the university’s resources, ensuring contextual relevance and accuracy tailored to student queries.

- **Training Set:** The majority of curated pairs were allocated to the training set, enabling the model to learn patterns and relationships across diverse query types, including academic, administrative, and departmental information.
- **Testing Set:** A smaller portion of curated pairs was reserved for testing, including complex and edge-case queries. This set was used exclusively for evaluating the system’s final performance in generating accurate and relevant responses.

The split was carefully designed to include a wide variety of query types, ensuring the system’s

robustness and ability to handle real-world scenarios effectively.

## 4.2 Implementation

The implementation leverages multiple advanced tools and libraries to ensure efficient retrieval and accurate response generation:

**Hugging Face:** Hugging Face provides a comprehensive repository of pre-trained models, including Qwen-2.5-14B, which was employed as the base language model for this project. Hugging Face’s ecosystem facilitated smooth integration and fine-tuning of the model for our use case. *Hugging Face website:* <https://huggingface.co/Qwen/Qwen2.5-Coder-14B-Instruct>

**Qwen-2.5-14B:** The Qwen-2.5-14B model served as the core for language understanding and response generation. This model demonstrated superior capability in handling complex queries, making it well-suited for our chatbot’s requirements. It was integrated to dynamically generate responses based on retrieved contexts, ensuring relevance and accuracy.

**LoRA (Low-Rank Adaptation):** We utilized LoRA to fine-tune the Qwen-2.5-14B model efficiently. LoRA updates a subset of model parameters, thereby reducing memory usage while achieving performance improvements. This approach allowed fine-tuning of large models like Qwen-2.5-14B on constrained computational resources.

*Citation:* Hu et al. (2021), *LoRA: Low-Rank Adaptation of Large Language Models*. For more details, see the full paper: <https://arxiv.org/abs/2106.09685>

**FAISS (Facebook AI Similarity Search):** FAISS was employed for similarity-based retrieval of embedded text chunks. It enabled efficient real-time vector searches to retrieve the most relevant contextual data for user queries, forming the backbone of the Retrieval-Augmented Generation (RAG) pipeline.

*Citation:* Johnson et al. (2019), *Billion-scale similarity search with GPUs*. For more details, see the full paper: [https:](https://)

[//arxiv.org/abs/1702.08734](https://arxiv.org/abs/1702.08734)

**SentenceTransformers:** The E5-large-v2 embedding model from SentenceTransformers was used to generate dense vector representations for both user queries and preprocessed text chunks. These embeddings were critical for robust and contextually accurate retrieval.

*Citation:* Wang et al. (2022), *Text Embeddings by Weakly-Supervised Contrastive Pre-training*. For more details, see the full paper: <https://arxiv.org/abs/2212.03533>

**GMU Hopper Cluster:** For performing model training, fine-tuning, and inference tasks, we utilized George Mason University’s Hopper Cluster, leveraging the available A100 GPUs (both 40GB and 80GB versions). <https://orc.gmu.edu/>

### 4.3 Results

The chatbot was tested with multiple queries to evaluate its ability to retrieve accurate and relevant information. The results demonstrate that the chatbot effectively utilizes the embedded data and FAISS-based retrieval to respond with precise and contextually correct answers. Below are some of the successful responses for simple queries:

- **Query:** What is the computer science department mailing address?

**Response:** 4400 University Drive, MSN 4A5, Fairfax, VA 22030, USA

- **Query:** How do I contact Antonios Anastasopoulos?

**Response:** Email: antonis@gmu.edu, Phone: 703-993-6821, Office Location: ENGR 4412, Office Hours: Mondays 3:00 PM - 3:30 PM. To schedule an appointment, send an email. Webpage: cs.gmu.edu/antonis

- **Query:** Where is the undergraduate advising hall located?

**Response:** Buchanan Hall Room D215

- **Query:** Who are part of the undergraduate advising team?

**Response:** Linda Sheridan, Joshua Fletcher, Vernel Wilks, Kristi Morro

The above instances showcase the chatbot’s ability to retrieve the exact information and pro-

vide accurate responses to user queries. This highlights the successful integration of the retrieval-augmented generation (RAG) approach in processing queries and generating relevant answers.

Additionally, screenshots of the running chatbot interface were captured to demonstrate the working system. Figure 2 shows the webpage with the chatbot open, while Figure 3 depicts the chatbot providing a correct response to a user query.

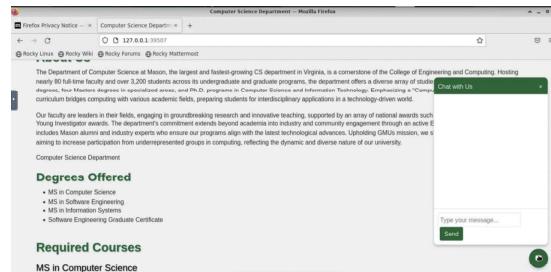


Figure 2: Webpage with the chatbot interface.

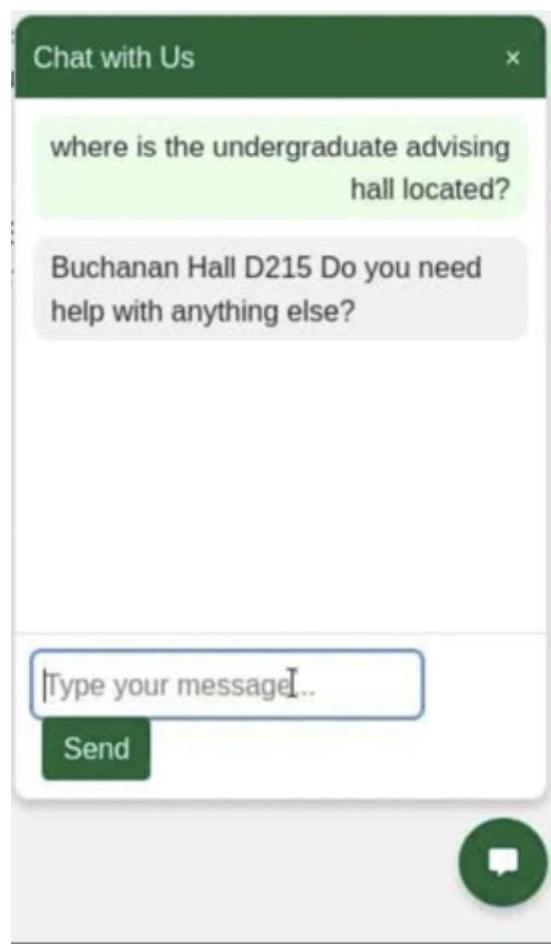


Figure 3: Chatbot answering a user query.

#### 4.4 Discussion

**Data Availability** One of the significant challenges encountered was the limited availability of pre-structured datasets. A considerable portion of the data was manually curated to ensure proper formatting for effective retrieval. This process, although labor-intensive, was crucial to provide a structured hierarchy for FAISS indexing and accurate context extraction. To address this, we implemented hierarchical labeling and contextual chunking strategies that enhanced the retrieval quality. However, as we scale the dataset, we anticipate the complexity of maintaining uniformity and relevance to increase.

**Fine-Tuning and Model Adaptation** For fine-tuning, we utilized LoRA (Low-Rank Adaptation) to efficiently adapt the Qwen-2.5-14B model. While LoRA significantly reduced computational overhead and memory usage, the observed performance improvements were moderate, especially for complex queries. This limitation stems from the lightweight nature of LoRA updates, which only impact a subset of the model’s parameters. As the dataset expands, we are considering hybrid approaches, such as integrating LoRA with partial fine-tuning of critical layers, to achieve more substantial performance gains while balancing resource constraints.

**Model Performance and Query Handling**  
The model demonstrated strong performance on straightforward queries, accurately retrieving and generating responses for single-context questions. However, it struggled with multi-faceted or ambiguous queries requiring reasoning across multiple chunks of context. These challenges highlight the limitations of the current retrieval-augmented generation (RAG) pipeline.

To address this, we attempted to enhance retrieval using re-ranking. However, embedding issues arose due to the 512-token limit of the embedding model, which caused semantically similar data to be scattered across different chunks, limiting the effectiveness of re-ranking. To overcome this, we plan to integrate a more advanced embedding model with a higher token limit to ensure better semantic grouping of related data. Additionally, improving retrieval and re-ranking scoring mechanisms will be prioritized to enhance the system’s ability to handle complex

queries.

**Future Improvements** Moving forward, we aim to address the discussed challenges by introducing:

- Dynamic context expansion to ensure all relevant information is included without exceeding token limits.
- Choosing better LLM with higher parameters for example Qwen-2.5-72B.
- Experimentation with model ensembles to combine the strengths of multiple architectures for improved performance on diverse query types.

These enhancements are expected to significantly improve retrieval quality, response accuracy, and overall user satisfaction.

#### 4.5 Resources

**Computation:** We are using GMU’s Hopper Cluster to handle our computational requirements. The cluster provides access to NVIDIA A100 40GB and 80GB GPUs for model fine-tuning and embedding tasks. LoRA fine-tuning, due to its efficiency, was completed in less than an hour, but we expect this time to increase as we continue expanding the dataset.

**Development Effort:** The largest time investment has been the manual structuring of data and creating ground truth references for validation and evaluation. We are exploring ways to optimize this process as the project progresses.

**Team:** The project is being developed by a team of three students, each of us responsible for different aspects of the project, allowing simultaneous progress in multiple areas.

**Cost:** The computational resources for this project were provided through GMU’s Hopper Cluster account, which is free to use for students. This support has enabled us to perform complex tasks like model fine-tuning and data embedding without incurring additional costs.

#### 4.6 Error Analysis

Before presenting the error analysis results, we examine a few instances where the system demonstrated limitations. Each response highlights a

specific error type and its potential cause, followed by areas for improvement.

#### **Response with Incorrect Context:**

*Query: Is CS 530 a core course?*

**Qwen's Response:** Yes, CS 530 is a core course. It is listed under the core courses for the MS in Computer Science degree requirements. The course title is Mathematical Foundations of Computer Science. It covers basic mathematical structures, mathematical logic, and probability theory with applications to problem-solving and formal reasoning, including hands-on practice using computational tools. Note that it may be taken for credit by students in the PhD CS program who did not go through the Bachelors-Accelerated Masters pathway. However, the primary context indicates its status as a core course for the MS program. To meet the MS CS general requirements, at least four advanced courses (12 credits) must be chosen, but CS 530 is specifically mentioned as a core course.

**Analysis:** The response retrieved an incorrect context by conflating core courses with general degree requirements. This highlights a retrieval failure where the retrieved chunk did not align precisely with the query's intent.

#### **Response with Embedding Failure:**

*Query: Can you tell me basic courses under systems and networks area?*

**Qwen's Response:** Computer Systems Fundamentals (CS 555), Computer Communications Networking (CS 571), Operating Systems (CS 587), Introduction to Cryptography (ISA 562), Information Security Theory and Practice (ISA 564) are listed as basic courses under the systems and networks area. Note that there's a typo in the original text for "basi" which should be "basic". However, it does not affect the course listings. The course names and codes provided are the ones available in the given context. There is an inconsistency with "Introduction Cryptography" being listed without "to", but based on the context, it seems to be a typo and should be "Introduction to Cryptography." These corrections...

**Analysis:** The embedding failed to group semantically similar course descriptions, leading

to incomplete and scattered retrieval. This issue is primarily due to the limited token length of the embedding model, which fails to cluster related data effectively.

#### **Response Limited by Context Length:**

*Query: How do I contact the computer science department?*

**Qwen's Response:** You can contact the Computer Science Department at George Mason University by calling 703-993-1530, emailing csinfo@gmu.edu for administrative support, csphd@gmu.edu for PhD program inquiries, esgrad@gmu.edu for MS programs, and undergrad\_programs for undergraduate programs. For physical visits, Graduate Advising is located in the Engineering Building, Room 4300, and Undergraduate Advising is in Buchanan Hall, Room D215. Visit <https://cs.gmu.edu/about/visit-the-department> for maps and directions. The mailing address is Department of Computer Science, George Mason University, 4400 University Drive, MSN 4A5, Fairfax.

**Analysis:** The response was trimmed due to context length limitations, omitting crucial information about advising and department details. This issue arises when the retrieved chunks exceed the model's input token limit, leading to context truncation.

#### **Response to Complex Query Leading to Failure:**

*Query: Can you tell me about CS 531 and is it a mandatory course?*

**Qwen's Response:** CS 531 is titled Computer Systems Fundamentals with a focus on systems programming. It is listed under the computer science primary course list and has a lecture schedule type. The registration restrictions are not specified as mandatory for all students but are limited to students within the engineering and computing college. Mandatory status is not indicated in the provided context. To determine if it's mandatory, refer to specific degree requirements or concentration descriptions which are not fully detailed here. Based on the given information, it does not specify that CS 531 is a mandatory course. However, it could be part of

certain concentrations or elective choices within the curriculum. For precise details, consult the complete degree requirements or course catalog. Given the context, it's not explicitly stated as mandatory.

**Analysis:** The model struggled to handle the multi-faceted query, providing vague and incomplete details. This highlights the limitations of reasoning across multiple chunks and accurately addressing nuanced questions.

## 5 Conclusion

In this project, all three team members contributed to data collection.

**Nitesh** was responsible for preprocessing the data, preparing the structured dataset, and creating the testing split for the model.

**Santhosh** handled the fine-tuning of the model and prepared the training portion of the fine-tuning data.

**Rohit** focused on testing, evaluation, and preparing the validation split for the data.

Additionally, all three team members worked closely together, supporting one another's responsibilities to ensure effective collaboration and teamwork. All of us participated in evaluating the model's performance using ground truth data, and collectively worked on optimizing the model to achieve the desired results.

## References

Hussam Abdulla, Asim Mohammed Eltahir, Saleh Al-wahaishi, Khalifa Saghair, Jan Platos, and Vaclav Snasel. 2023. Chatbots development using natural language processing: A review. *IEEE Access*, X:10017592.

Sumit Kumar Dam, Choong Seon Hong, Yu Qiao, and Chaoning Zhang. 2024. A complete survey on llm-based ai chatbots. *arXiv preprint arXiv:2406.16937*.

Kenneth McGarry Guendalina Calderini, Sardar Jaf. 2023. A literature survey of recent advances in chatbots. *MDPI*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih,

Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2004.04906*.