

# Carolinian Chatbot: End-to-End Question-Answer Pipeline on the University of San Carlos Policies, Procedures, and Guidelines Using Retrieval Augmented Generation

Wayne Matthew A. Dayata

*Department of Computer, Information Sciences, and Mathematics*

*University of San Carlos*

*Cebu City, Philippines*

wdayata@gmail.com, 20100215@usc.edu.ph

**Abstract**—The recent advancements in document embedding and context-based inference models have fueled the development and deployment of question-answering (QA) pipelines. This paper addresses a common challenge faced by universities: student inquiries regarding policies, procedures, and guidelines that often overwhelm staff and lead to delays. To address this, we present a novel End-to-End QA application employing the Retrieval-Augmented Generation (RAG) strategy. Our system leverages the University of San Carlos Student Manual 2023 and supplementary documents, along with the Nomic-Embed-Text-v1.5 embedding model and Meta’s powerful LLaMA-3 inference model. Deployed on Streamlit Cloud, the application allows students to access initial responses to their questions. User interactions and retrieved data indicate reliable performance in both document retrieval and inference phases, particularly for queries spanning multiple documents. These results demonstrate the application’s potential to handle common student inquiries, thereby reducing workload on university staff and personnel.

**Index Terms**—Natural Language Processing, Question Answering, Retrieval Augmented Generation, Document Embedding, Vector Search, Large Language Models

## I. INTRODUCTION

This section of this paper discusses the background of the study and describes the current context, problems, and relevance that led to the motivation for developing this project.

### A. Rationale

The digital age has ushered in an era of information overload, presenting challenges across various domains. In the context of higher education, college students often struggle to navigate the vast amount of text-based information related to their academic journey. University handbooks, enrollment guidelines, and office directories can be overwhelming, especially for students navigating a large campus with numerous departments and resources. Locating the right information quickly and efficiently becomes crucial for completing tasks like enrollment or acquiring specific university services. With the recent increase in student population, officers of student organizations as well as staff from the offices are also increasingly overwhelmed with the influx of student queries which

most of the time were not addressed to the right personnel. The students’ lack of adequate knowledge about their university’s guidelines and rules results in the lack of timely access to and guidance from their office of concern and the underutilization of the student support services offices and facilities. These facts and issues call for a more efficient approach to information retrieval for students to receive accurate and timely information about their school concerns.

Fortunately, advancements in question-answering (QA) and Natural Language Processing (NLP) offer promising solutions to this information overload problem [1]. QA systems, powered by NLP techniques, can function as intelligent virtual assistants [2]. These systems, trained to understand natural language queries, can retrieve relevant information from college handbooks and university websites. Imagine a chatbot system accessible to students 24/7, capable of answering questions like “What are the requirements for enrollment?” or “Where can I find the scholarship office?” Such a system would empower students to navigate university procedures independently, alleviating the burden on student representatives and faculty who are often inundated with repetitive inquiries.

NLP research has blossomed into a dynamic field, constantly pushing the boundaries of human-computer interaction. Beyond question answering, NLP finds applications in various real-world scenarios. Businesses leverage sentiment analysis of social media data to understand customer preferences [2]. Machine translation tools, powered by NLP, break down language barriers and foster global communication. Automated document summarization streamlines information processing for researchers and professionals alike.

Looking ahead, the integration of NLP into chatbots and virtual assistants will continue to transform how we interact with technology. As NLP models like LLAMA 3 demonstrate, the ability to process and understand natural language with ever-increasing sophistication opens doors for a future where technology seamlessly integrates into our daily lives, offering intelligent support and guidance [3].

## B. Objectives and Scope

Having been inspired by the increased potential and relevance of NLP solutions in real-world contexts, the author sought to contribute further by developing the Carolinian Chatbot: a chatbot application utilizing a question-answer pipeline that can generate accurate and sound responses from the context of the University of San Carlos (USC) Student Manual and its supplementary documents. The specific objectives, which will be elaborated in this paper, are as follows:

- 1) Preprocess and analyze the relevant texts as inputs for the pipeline.
- 2) Visualize the retrieval of relevant documents for the common questions.
- 3) Build the question-answering chain with the LLaMA-3 model and its accompanying application.
- 4) Deploy the application online with Streamlit.
- 5) Evaluate the overall pipeline.

The development and use of the chatbot are subjected to the following conditions and restrictions:

- 1) Only open-source embedding and large language models will be used.
- 2) As this study focuses only on RAG, no fine-tuning will be performed.
- 3) The answers generated from the pipeline will be strictly within the given context, thus providing info updated only as of the time the texts are retrieved for the model.
- 4) The chatbot cannot be used beyond the scope of the USC related contexts.

## II. TECHNICAL BACKGROUND

This section elaborates on the fundamental concepts and technical terms applied throughout this study. The descriptions cover techniques in the field of NLP, specifically on RAG.

### A. Comparison of LLM Strategies

There are several methods for adapting large language models for performing specific tasks, whether it may be question-answer, summarization, classification, or any other completion tasks. These methods can be summarized into three general categories - prompt engineering [4], fine-tuning [5], and retrieval augmented generation (RAG) [6]. Fig. 1 summarizes the benefits of each technique discussed.

- **Prompt engineering** emerges as a crucial technique for maximizing the efficacy of large language models (LLMs). It emphasizes the importance of meticulously crafting the input prompts or examples provided to LLMs. This meticulousness stems from the recognition that the phrasing and structure of these prompts significantly influence the nature of the model's generated outputs and overall behavior.

- **Fine-tuning** is a transfer learning technique that involves further training a pre-trained language model on a specific task or dataset. The pre-trained model, which has already learned general language patterns and knowledge, serves as a solid foundation. During the finetuning process, the

model's weights are adjusted to better suit the target task or domain, effectively specializing the model for that particular use case. It can be more computationally efficient than training a model from scratch, but can be highly difficult for the model to improve and adapt effectively if the data provided has a domain that is significantly different from that of the pre-trained data.

- **RAG** is a technique that combines the generative power of language models with the ability to retrieve relevant information from external data sources, either online or offline and compiled by the user. The core idea behind RAG is to leverage the vast amount of knowledge encapsulated in these data sources to enhance the language model's outputs, making them more informative, factual, and grounded in real-world knowledge. RAG has proven effective in tasks such as open-domain question answering, where the ability to access and incorporate external knowledge can significantly improve the quality and accuracy of the generated responses while also avoiding hallucinations or the tendency of the model to generate false information.

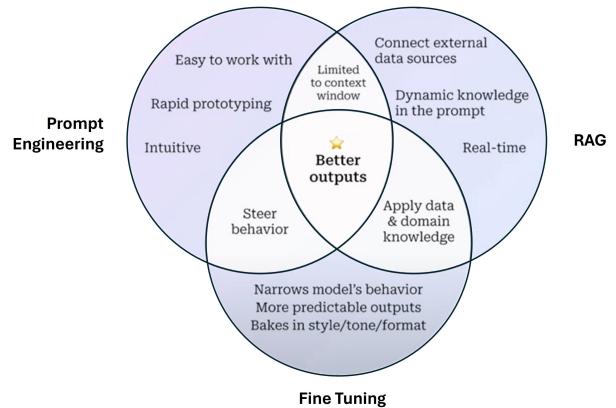


Fig. 1. Behavioral advantages of LLM strategies

Fig. 2 presents bar charts comparing the accuracy, cost, implementation complexity, and flexibility [7].

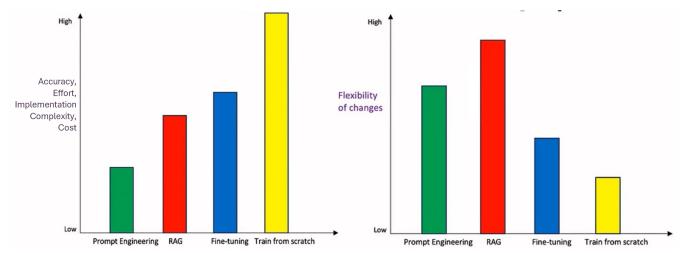


Fig. 2. Accuracy, cost, and flexibility of LLM strategies

Among the techniques, training a model from scratch yields the highest possible accuracy, followed by fine-tuning and RAG, and lastly purely prompt engineering, while for flexibility, this is led with RAG, followed by prompt engineering

and fine-tuning, and training from scratch [7]. It is important to note that both fine-tuning and training LLMs from scratch require a well-curated dataset that can be directly used for additional training, which is rare to be seen and is very computationally expensive. On the other hand, RAG requires much fewer resources for the retrieval and inference phases while still achieving reasonable accuracy levels, which makes it ideal for most use cases including this project.

### B. Text Embedding

A fundamental hurdle in Natural Language Processing (NLP) lies in establishing a bridge between the nuanced complexities of human language and the numerical representations computers utilize. Text embedding techniques emerge as a critical solution, transforming textual data – words, sentences, or documents – into numerical vectors. These vectors encapsulate the semantic and syntactic properties of the text, empowering machines to process and analyze language with greater efficacy [8].

Consider a vast library, where each document represents a book and individual words function as unique entries. Traditionally, computers would struggle to grasp the relationships between these words. Text embedding offers a novel approach. Words that share semantic closeness, such as "king" and "queen," are positioned in proximity within a high-dimensional space. Conversely, words with no inherent connection, like "king" and "banana," reside farther apart. This method grants machines the ability to not only recognize individual words but also grasp the inherent interplay of meaning and connections within textual data. Fig. 3 illustrates this through arbitrary embeddings for words and documents.

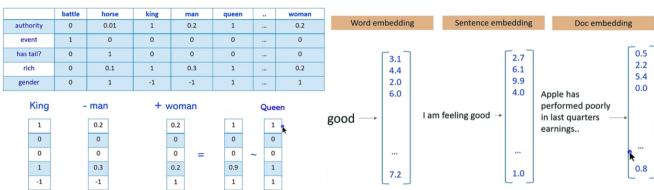


Fig. 3. Examples of text embeddings

Because word embeddings contain several dimensions and can no longer be visualized and spatially represented in a human-compatible and understandable form, dimensionality reduction needs to be performed prior to visualization. These dimensionality reduction techniques include PCA, t-SNE, and UMAP, where the latter of these was used by the author in the examination of the generated embeddings [9].

Within the realm of text embedding models, Nomic-embed-text stands out as a popular open-source option designed to address the intricacies of long-context text. Unlike earlier models with limitations in handling extended passages, Nomic-embed-text boasts a robust architecture capable of processing sequences stretching up to 8,192 tokens. This extended capability proves particularly valuable for tasks that necessitate

comprehension of nuanced relationships within lengthy documents, such as question answering or summarizing complex research papers. According to the technical report, Nomic Embed outperformed other closed-source models on both short and long-context tasks on popular dataset benchmarks such MTEB and Loco despite having smaller parameter count and memory requirements (around 0.5 GB) than most other models need. Fig. 4 presents a comparison of Nomic Embed's performance on the mentioned benchmarks with that of other embedding models.

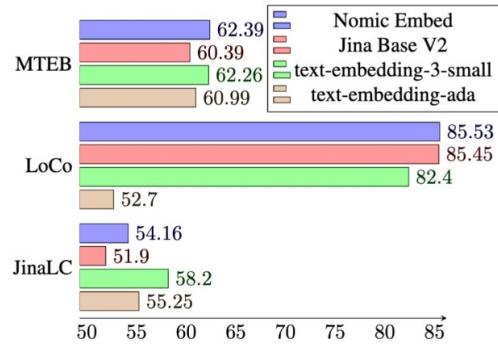


Fig. 4. Comparison of Nomic-Embed with other Embedding Models

Furthermore, Nomic-embed-text prioritizes transparency and auditability, a crucial aspect for NLP applications. The model's creators have made both the training data and model weights publicly available. This openness allows researchers and developers to not only utilize the model but also meticulously examine its inner workings, fostering trust and enabling further refinement [10] [11].

### C. Vector Databases

Vector databases have become increasingly crucial across various fields due to their exceptional ability to handle high-dimensional data, often referred to as vectors. These specialized databases are designed to efficiently store, index, and facilitate rapid searches within this complex data structure. Each piece of information within a vector database is represented as a vector, residing in a multi-dimensional space. This flexibility allows vectors to capture a diverse range of information, including numerical features, text and image embeddings, and even intricate data representations like molecular structures [12]. Fig. 5 illustrates the splitting of document into chunks with each being generated an embedding and placed together with the chunk into a vector store.

The core functionalities of vector stores are as follows [12]:

- 1) **Indexing:** After a vector embedding is stored within the database, a variety of algorithms come into play. These algorithms work together to map the embedding onto data structures that enable faster searching. This process, often referred to as Approximate Nearest Neighbor (ANN) search, can utilize techniques like hashing, graph-based search, or quantization. These techniques

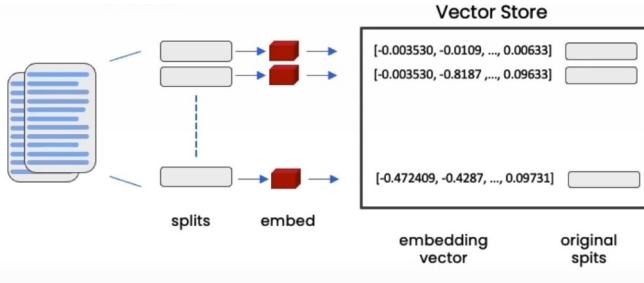


Fig. 5. Vectore store example

are combined into a pipeline to retrieve similar vectors (neighbors) based on a user's query.

- 2) **Querying:** Once the search process is initiated, the vector database compares the user's query vector to the indexed vectors. It then employs a similarity metric to identify the nearest neighbors to the query vector.
- 3) **Post-Processing:** The final step may involve post-processing, depending on the specific vector database used. This post-processing stage refines the retrieved nearest neighbors, potentially re-ranking them for future reference, and ultimately producing a final output for the user's query.

ChromaDB is an open-source vector store designed for storing and retrieving vector embeddings [13]. Its primary function is to store embeddings and associated metadata for extensive language models to utilize later. Additionally, it can be employed for semantic search engines that operate on text data. ChromaDB offers the unique ability to store vectors alongside additional metadata, allowing for filtering during the search process. This database leverages SQLite and provides users with the option of in-memory storage or local persistence. Notably, ChromaDB is compatible with various embedding models, and for this specific project, the Nomic-Embed model (discussed in Section II-B) will be used.

#### D. Document Search and Retrieval

During document retrieval, a query is also transformed into a vector. Similarity search techniques then compare this query vector to the indexed document vectors. By analyzing the distances between these vectors, these techniques can identify documents that share the closest semantic meaning to the query. This enables the retrieval of highly relevant documents, even for complex or nuanced queries, significantly improving the efficiency and accuracy of information retrieval tasks.

The four main semantic search techniques can be summarized as follows [12]:

- **Manhattan distance:** This computes the sum of absolute differences between corresponding elements of two vectors, giving the distance formed with horizontal and vertical lines which ignores directionality in vector space. This is mostly used in misspelling corrections.

- **Euclidean distance:** This computes the straight-line distance between two points in vector space. While being more intuitive to distance measures, it can be sensitive to magnitude differences. This is mostly used in general document similarity searches.

- **Dot product:** This measures the directional similarity between two vectors. Though being very computationally efficient, it is highly dependent on the magnitude. However, this is useful if the order of words is not crucial, therefore not making the best choice if semantic meanings are to be accounted for.

- **Cosine similarity:** This computes for the cosine of the angle between two vectors projected in a multi-dimensional space, though requiring more complex computations than the rest, it is the most ideal technique for finding document similarity according to semantic meaning as this is focused over the wording differences.

Fig. 6 illustrates these distance metrics used in performing similarity search, along with their mathematical formulas.

### Distance Metrics in Vector Search

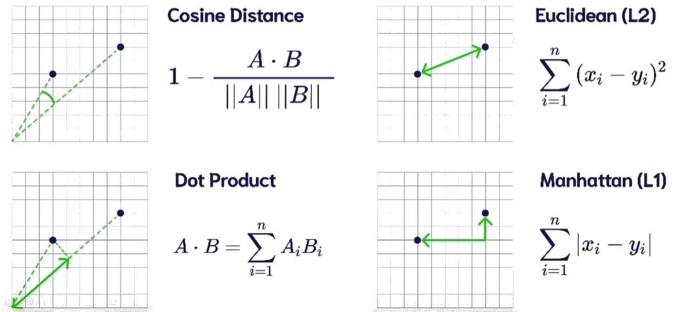


Fig. 6. DIstance metrics in vector search

#### E. Transformer and LLaMA Architecture

The transformer architecture processes language through an encoder-decoder structure [14]. The encoder analyzes the input sequence, breaking it down and understanding relationships between words using self-attention. Self-attention allows the model to focus on relevant parts of the sequence for understanding each element. The decoder then uses the encoded information to generate the output text sequence, one word at a time, again using attention layers to focus on relevant parts of the encoded data for each generated word. This parallel processing and ability to capture long-range dependencies make transformers fast and effective for various NLP tasks.

Although the transformer architecture has revolutionized natural language processing (NLP) tasks, recent advancements have introduced LLaMA [15], a powerful contender. Both architectures rely on encoder-decoder structures, processing input text via encoding and generating output via decoding. However, LLaMA incorporates several architectural improvements upon the transformer. Fig. 7 shows a visual comparison of the two architecture [16].

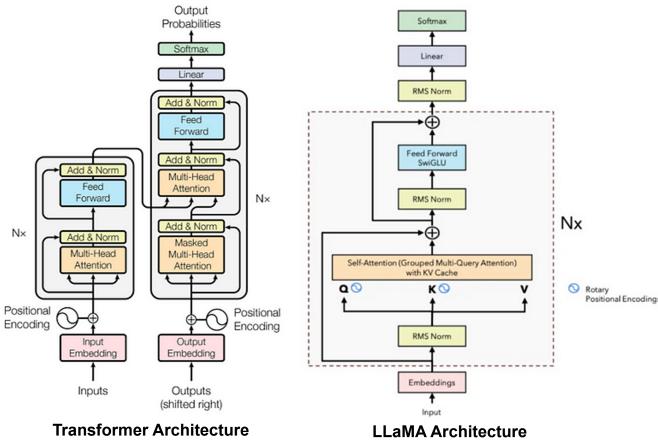


Fig. 7. Transformer architecture vs. LLaMA architecture

The following summarizes the changes implemented in the LLaMA architecture [16] [17]:

- 1) **Normalization:** While transformers use layer normalization after each sub-layer, LLaMA employs Root Mean Square (RMS) normalization, improving training stability.
- 2) **Activation function:** While transformers rely on the ReLU activation function, LLaMA utilizes SwiGLU which is known for better performance.
- 3) **Positional encoding:** While transformers employ absolute positional embeddings, LLaMA leverages rotary positional embeddings (RoPE), a process allowing the model to learn dynamic positional representations during training rather than relying on pre-calculated static positional encoding vectors, reducing parameter usage and potentially improving efficiency.

This project utilizes the 8B parameter variant of the LLaMA-3 model for inferencing. This model, developed by Meta, leverages a colossal training dataset exceeding 15 trillion tokens. This represents a significant advancement compared to previously employed datasets, potentially enabling LLaMA 3 to achieve refined comprehension of language intricacies and propel performance on diverse NLP tasks.

### III. METHODOLOGY

This section explains the datasets, tools, and procedures needed to gather and perform to achieve the project objectives. This includes the conceptual framework, a discussion of the pipeline's components, and means of deploying the project.

#### A. Conceptual Framework

The main goal of this project is to develop an End-to-End question-answer pipeline from the USC resources. Fig. 8 shows the conceptual framework of the study which shows the specific steps involved and the flow of the components. The detailed phases in the Fig. are elaborated in the remainder of the section.

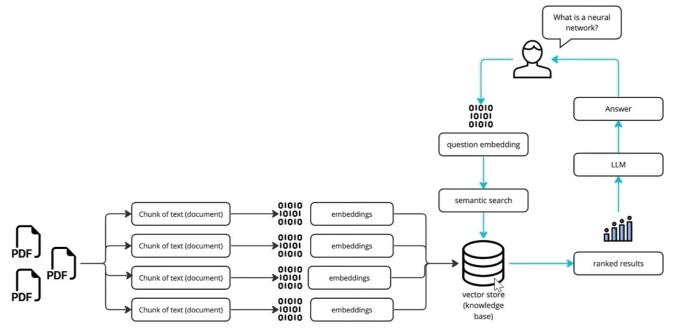


Fig. 8. Conceptual Framework

#### B. Data Sources and Preprocessing

The texts used for the context of the chatbot are the following:

- 1) USC Manual for Undergraduate Students, 2023 Edition [18]
- 2) USC Enrollment Guide as of first semester AY 2023-2024 [19]
- 3) USC Directory of Student Support Offices and Program Owners [20]
- 4) Guidelines for Motor Vehicle Pass Stickers found in the university's Integrated School Management Information System (ISMIS)
- 5) FAQ posts from the USC Supreme Student Council and USC-Computer and Information Sciences Council Facebook pages highlighting commonly asked enrollment related concerns
- 6) Repository of the undergraduate program brochures found on the USC website's Academics page [21]

Sources (1) and (6) are in PDF formats, source (3) is in Google Slides presentation format, while sources (2), (4), and (5) are in web formats viewable online. However, source (4) can only be visible to students with ISMIS accounts as the viewers are only restricted to official USC students, admin, and staff.

The preparation of the text files are mostly done manually instead of the usage of libraries that automatically scrape or extract files from other file formats. This is to eliminate the formatting and typographical errors found in certain parts of the document. There were cases where the orders of information in a bulleted list or table were disarranged when copied or extracted, prompting the need to manually input and re-order the contents. Sections from some undergraduate program brochures could not be selected and extracted automatically, thus an OCR tool was used to detect and extract text from the affected section. For all the documents, Fixes in capitalization, punctuation, and spacing were made to remove unsupported characters, as well as unnecessary spaces between a word to prevent models from misinterpreting those occurrences as two separate tokens which can potentially alter the meaning of the relevant context.

Certain sentences were also improved and rewritten to provide clearer contexts, especially for those that are commonly targeted, such as enrollment-related texts. For example, the steps to access a certain module in the university's ISMIS portal regarding fees, assessments, and promissory notes, which were not stated in any of the sources, were manually added. Abbreviations were also gathered throughout the text which were placed together with its equivalent meanings in a CSV file, which is then used to generate sentences per pair of abbreviation and meaning, making the model easy to access and answer abbreviation-meaning questions. Lastly, paragraphs with common contexts were transferred to a merged text file and placed adjacent to each other such that they can remain together when the documents are divided into chunks in the next preprocessing phase.

### C. Pipeline Construction

1) *Vector Database Creation:* At this phase, it is expected that the articles have been reprocessed as discussed in Section III-B and stored in different text files, with each text file representing an identifiable subtopic that can be queried for most possible questions involving the subtopic. Note that they were not combined into one big text file to avoid the appearance of paragraphs from multiple topics in a single chunk. For each article or text file, we apply LangChain's [22] `RecursiveCharacterTextSplitter()` method to divide the article into document chunks with a specified maximum token limit (with the computation of tokens facilitated by tiktoken's "cl100k\_base" encoding function) and a specified maximum token overlap limit between chunks. The said method first splits texts by paragraph through `\n\n` tokens, followed by newline `\n` and period characters, and lastly by space characters. This results in each article containing at least one chunk to multiple chunks depending on the length of the text.

Next, each document chunk is then fed into the Nomic-Embed-Text embedding model which is another open resource obtainable in the HuggingFace platform [23]. This allows each document to be encoded with vectors of 768 dimensions with each dimension representing a certain semantic meaning to distinguish the relevance of one context to another or a specific theme. These encoded vectors, together with the original document source and the metadata (name of the source text file) are then saved in a vector store using the ChromaDB library which allows the generated vector store to be persisted and reusable in the upcoming parts of the pipeline.

The items that can be tweaked at this phase are the maximum number of tokens and overlapping tokens per chunk, and the embedding model to be used. Note that one should take into consideration the number of chunks that they desire to retrieve for better flexibility while also avoiding the possibility of going beyond the context limit supported by the LLM.

2) *Retrieval Augmented Generation:* This phase involves the searching of relevant document chunks which will serve as the context for the LLM along with the question for it to

generate an appropriate answer based on further analysis of the contexts by the LLM. At first, the question string is fed into the embedding model to create an embedding vector with the same dimensions as that of the other document chunks. This embedding vector is then compared with the rest of the embeddings found in the vector store using the specified vector similarity metrics (refer to Section II-D) such as cosine similarity and Euclidean distance.

Once the metrics are computed for each embedding, the documents with the top scores are extracted from the vector store. Here, it is important to verify that the eventual answer to the question can be answerable within the given set of document chunks to infer that the correct documents were retrieved. This part can be dangerous if the maximum token length per chunk is too small since the entire context is too subdivided and cannot be fully extracted for questions that require the entirety of the subtopic like those that ask for descriptions and step-by-step procedures for a task.

A visualization of this process can be done through Uniform Manifold Approximation and Projection (UMAP) [24], a dimensionality reduction tool that can accurately map the vectors in a lower dimensional space that is interpretable to humans.

The document chunks are then passed together with the question to the Llama-3-8B model by wrapping the context and question in a well-crafted prompt template inside LangChain's [22] `RetrievalQACChain` pipeline, a class commonly used for retrieval question-answering tasks in NLP projects as this forces the model to base its answers only on the context and avoid generating other irrelevant information should the answer be not found in the context. As soon as the model reads through the texts and identifies the relevant tokens for the answer, these tokens are taken to generate the final answer to be returned to the program, which will finally be displayed together with the metadata of the document chunks indicating the source article topics.

The items that can be tweaked at this phase are the number of relevant documents to be returned in the retrieval phase, the similarity metric to be used, the LLM to be utilized to process the input documents and questions, the temperature of the LLM which controls the randomness, and the prompt template shall the developer wants it rewritten.

### D. Application Development and Deployment

At this phase, the author developed an interactive application utilizing the built RAG pipeline through Streamlit, a popular framework for creating interactive data science applications that can scale pipelines to their dynamic usage instead of leaving them stuck in Python scripts or Jupyter notebooks. The popularity of Streamlit is also increased by its free hosting capabilities in the Streamlit Community Cloud platform [25], which easily allows the application to be shared and used by others within the community but with a 1GB resource limit. Thus, this served as the reason for choosing an efficient embedding model in the initial phase of the pipeline.

For the author to evaluate the application based on the prompts made by other users, each question, answer, and source chunk entry was automatically stored in an online spreadsheet via Google Sheets, which is executed by passing the pair of texts through a POST request into a Google Apps Script that handles the data insertion.

#### IV. RESULTS AND ANALYSIS

This section illustrates the outcomes of the execution of the steps taken to achieve the objectives of the research study. Moreover, it expounds on the different observations of the results obtained as well as interpretations that suggest the effectiveness and improvement points of the project.

##### A. Exploratory Data Analysis

Before creating the pipeline, the author first analyzed the composition of the texts such as the distribution of chunks and the relevance from one chunk or topic to the other, as seen in Table 1. Here, the number of tokens is computed using the tiktoken library's "cl100k\_base" encoding function, which is also the basis of chunk splitting.

TABLE I  
DISTRIBUTION OF CHARACTERS, TOKENS, ARTICLES, AND CHUNKS  
ACROSS SOURCES

Source	Characters	Tokens	Articles	Chunks
Student Manual 2023	141,323	28,690	36	49
Enrollment Guide	51,493	10,521	22	24
Directory of Offices	35,048	7,700	4	11
Motor Vehicle Guide	7,927	1,553	2	2
FAQs from FB Pages	6,152	1,261	2	2
UG Program Brochures	112,657	20,284	42	42
<b>Total</b>	<b>354,600</b>	<b>70,009</b>	<b>108</b>	<b>130</b>

The bar plots for the distribution of tokens and articles per source are shown in Fig. 9, while the line plot of the character lengths and token lengths per article is shown in Fig. 10.

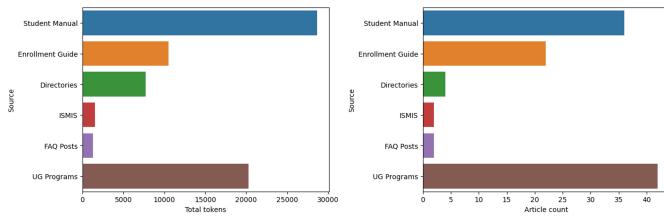


Fig. 9. Token lengths and article counts per source

In total, 95 of the 108 articles contained less than 1000 tokens, 7 of the articles contained between 1001 and 2000 tokens, and 6 articles contained 2000+ tokens. With this, it

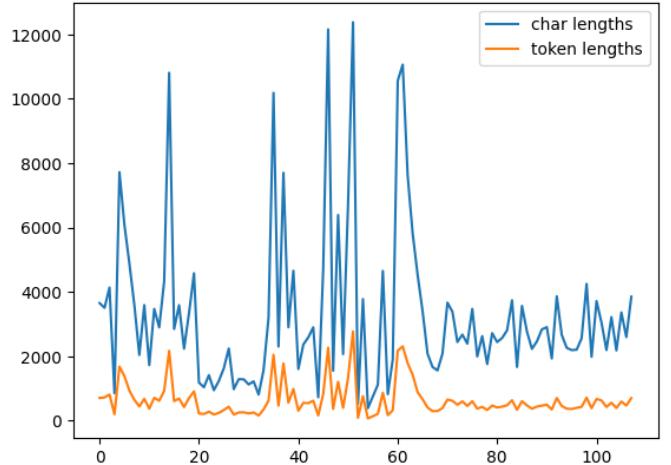


Fig. 10. Distribution of text lengths for all articles

would be ideal to set the maximum size for chunk splitting to 1000 tokens to preserve the full context of most smaller articles without compromising the coherence of paragraphs between articles. Despite having additional context references, the source with the most articles and tokens was still the Student Manual 2023, keeping it the project's primary focus.

##### B. Document Embedding

After the splitting of documents into chunks, the chunks were annotated with broader topics to be used as colors and labels in the embedding visualization. For the visualizations, the author performed dimensional reduction of the vectors via UMAP, resulting in the plots shown in Fig. 11 for all chunks, Fig. 12 for all chunks that do not belong to undergraduate programs, and Fig. 13 for all undergraduate program chunks.

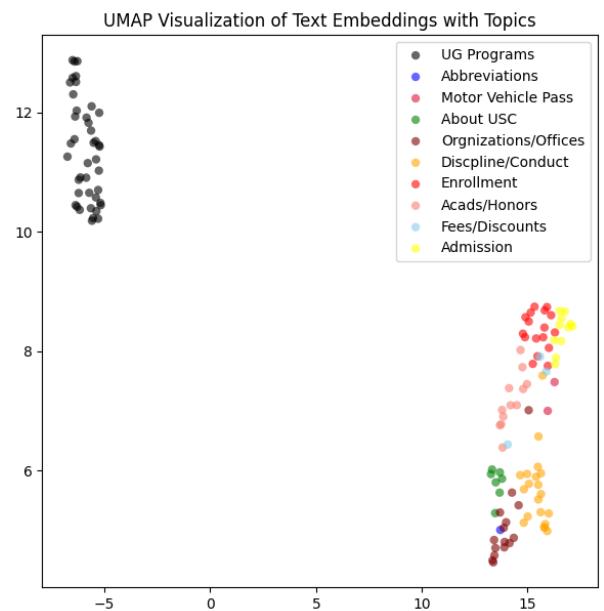


Fig. 11. Embeddings for all articles

In Fig. 11, it can be seen that the embeddings for the undergraduate program chunks are far apart from the embeddings of the other chunks. This tells that the structure of the undergraduate program brochures is completely different from that of the other articles containing procedural and imperative contexts. Also, the program chunks are mostly just an enumeration of the job positions and courses apart from a technical description which does not involve terms related to the enrollment guide or the student manual at all.

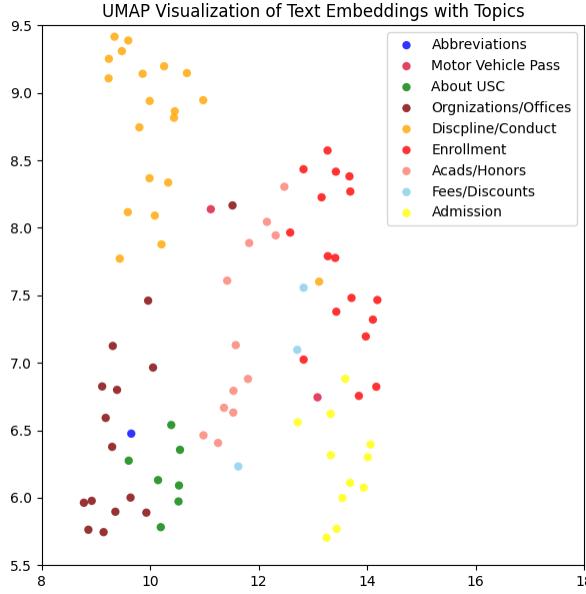


Fig. 12. Embeddings for articles except undergraduate programs

Zooming into Fig. 12, the annotated topics of each article were well-grouped together which validated the coherence of the contents of the sources. In addition, the closeness between two particular topic groups also suggested similarities between the different topic categories, such as admission with enrollment (since these are connected topics), About USC with abbreviations and organizations/offices (as the organizations and offices also introduce the image of USC), and enrollment with academics/honors (since part of the enrollment mechanics contains certain academic requirements).

The same arrangements and observations can also be seen across the articles on the undergraduate program brochures in Fig. 13 when grouped according to the schools offered, except for the SAS (dark blue - School of Arts and Sciences) whose points tend to be spread across three parts, suggesting some well integrations with other schools due to the similarity of certain course offerings, and so with SAFAD (maroon - School of Architecture, Fine Arts, and Design) whose points also spread across two parts. The top 3 blue dots for SAS - BS Computer Science, BS Information Technology, and BS Information Systems were closer to the SOE (School of Engineering) and SBE (School of Business and Economics), while the 6 blue dots for SAS represent the rest of the programs in the Sciences cluster which were well aligned with

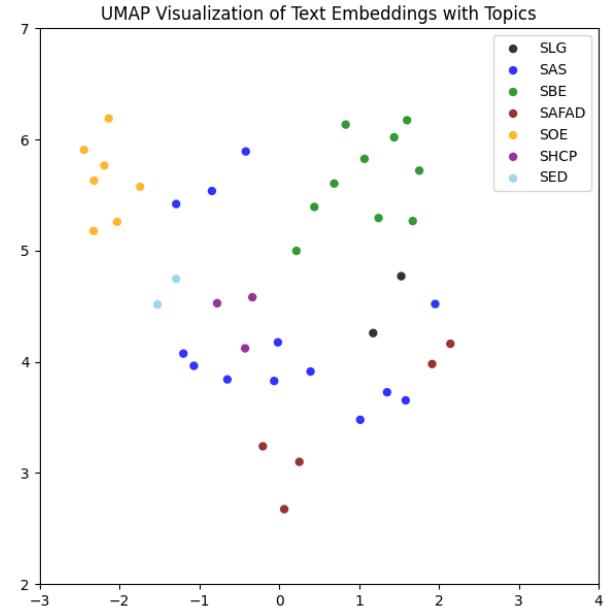


Fig. 13. Embeddings for undergraduate program articles

the SHCP (School of Health Care Professions) programs. The 2 maroon dots for SAFAD - the two majors of Fine Arts also blended well with the four blue dots representing the Arts cluster of SAS.

Through understanding the relevance between topics, it is now easier for the readers to trace the search of relevant documents in the retrieval phase due to the context similarity shown within the topic groups.

### C. Document Retrieval

This subsection evaluates the relevance of the retrieved documents towards the questions through UMAP visualization of the embeddings and the similarity scores between the embedding of the question and that of the retrieved documents. The author performed this analysis on seven questions which are all commonly asked by students and are open-ended. Figures 14 to 20 show the plots, relevant documents, and similarity scores per metric.

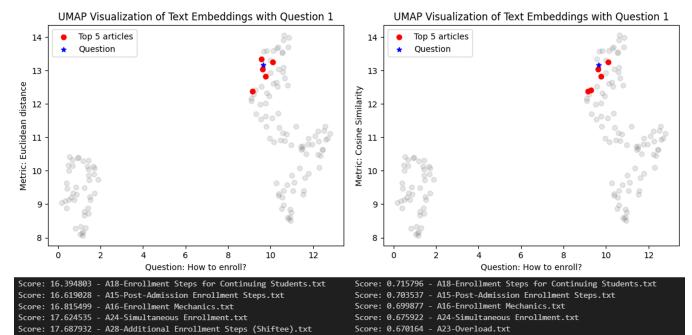


Fig. 14. Retrieved sources with embeddings for question 1

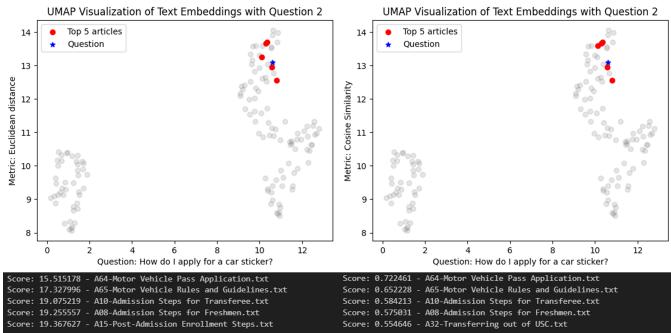


Fig. 15. Retrieved sources with embeddings for question 2

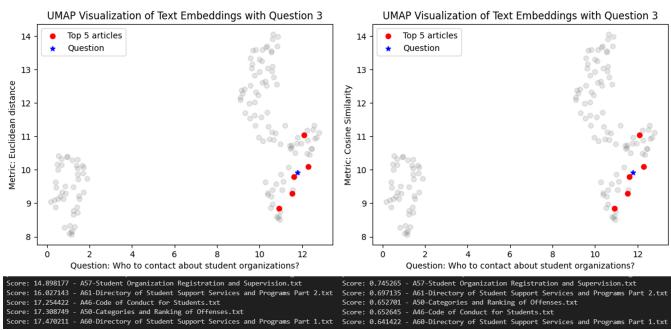


Fig. 16. Retrieved sources with embeddings for question 3

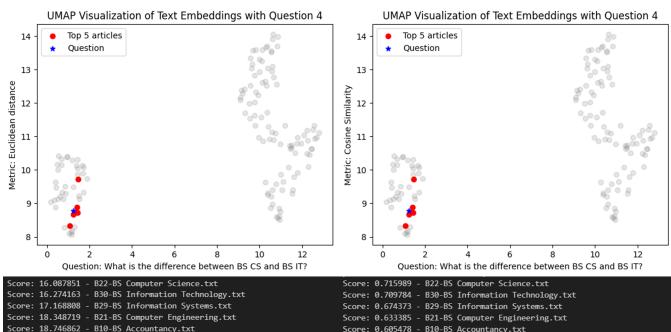


Fig. 17. Retrieved sources with embeddings for question 4

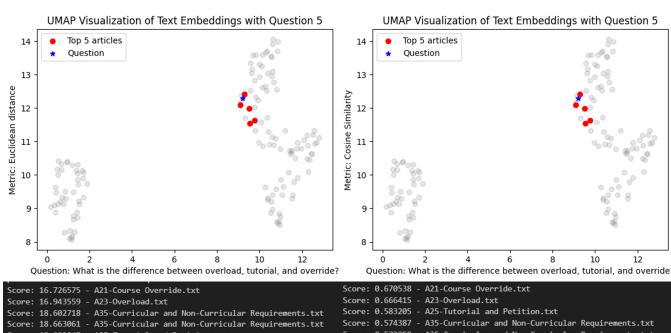


Fig. 18. Retrieved sources with embeddings for question 5

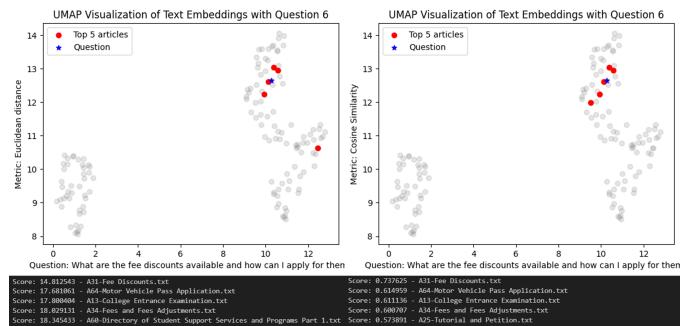


Fig. 19. Retrieved sources with embeddings for question 6

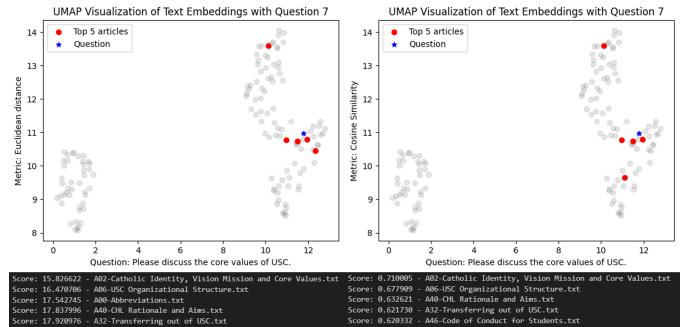


Fig. 20. Retrieved sources with embeddings for question 7

In the plots, the question embedding, retrieved document embeddings, and the other document embeddings are marked purple, red, and gray, respectively. The left plot shows the top 5 most similar chunks computed using Euclidean distance with their scores, while the right plot shows the top 5 most similar chunks computed using cosine similarity with their scores.

As seen in the figures, it can be seen that the retrieved questions are significantly close in all directions towards the question embedding, showing a superior ability of the embedding model to store meaningful information from each document as well as generate the proper embeddings for the questions to align with the similar articles. This also shows the importance of choosing a reliable and memory-efficient embedding model for document retrieval tasks.

Another notable observation from the figures is that in all seven questions, the top 2 most relevant articles based on Euclidean distance and cosine similarity metrics are the same; and that in 6 of 7 questions, the top 5 most relevant articles retrieved for both metrics are present, with the remaining question having the top 4 most relevant articles equal from both metrics. Lastly, the results also indicate that the document search performed well for complex question types such as comparative questions (Questions 4 and 5) which require references from multiple articles, questions with multiple answer variations (Question 1, which the model pulled multiple enrollment procedure articles for freshmen and continuing students), and questions from an article split to multiple chunks (Question 3) in which only the relevant chunk was retrieved out of three chunks. In question 5, the fifth

most relevant article held part of the key answer compared to the third and fourth relevant article where the terms were mentioned a few times. With this, it can be safe to set the maximum number of retrieved chunks (k) per question in the question answering chain to 5.

#### D. Sample Questions and Answers

With the large context window of 8192 tokens in the Llama-3 model for storing both the input and output tokens, there is enough room for the model to generate insightful text that addresses the questions asked based on the documents provided. Recall that the token limit per chunk was set to 1000 while the number of chunks for retrieval was set to 5, making the largest possible input token count 5000, or approximately 400-500 words spread across 3 paragraphs.

Fig. 21 shows Llama-3's responses to the sample questions which are all complete, accurate, and well-summarized from the sources.

**Who to contact about student organizations?**

For inquiries and concerns regarding student organizations, you can contact the Office of Student Formation and Activities (OSFA) via landline (032) 2300100 local 102/592, or through the following persons and emails:

- \* Student Activities Officer – Ms. Lizaneth Bustillo (lbusstillo@usc.edu.ph)
- \* Head of OSFA – Ms. Catherine Algoso (calsgoso@usc.edu.ph)

There are two fee discounts available:

1. \*\*Family and Sibling Discount\*\*. This discount is available for siblings enrolled in USC, excluding those in Montessori, Law, and Graduate Programs. The discount is 5% on tuition fees. To apply, click "Student Task", "Others", and "Apply Same Family Privilege" in the ISMIS portal. Indicate the academic period and year, enter the ID number of the sibling to be included in the privilege, and click "Submit". The application will be forwarded to the VP Finance for approval.
2. \*\*Full Payment Discount\*\*. This discount is available for students who pay their tuition fees in full. The discount rates vary depending on the payment method:
  - \* Offsite Payment Centers and Onsite Cash: 5% discount
  - \* Onsite Credit Card (Visa or Mastercard): 3% discount
  - \* Onsite Debit Card: 3.5% discount

To apply for the Full Payment Discount, contact the Accounting Office for verification and necessary computations.

Fig. 21. LLM responses to sample questions

The succeeding instances show some other notable observations encountered during the experiment phase. The first to be discussed is the impact of having too small chunk sizes to be detected for retrieval. This is shown in Fig. 22 where the enrollment steps retrieved are incomplete and only information from the beginning and end chunks are passed to the Llama-3 model and processed for answering.

**How do I enroll in USC?**

To enroll in USC, you need to follow these steps:

1. Submit the required documentation requirements at the Office of the University Registrar.
2. Make an initial payment of school fees.
3. Get a validated Study Load that authorizes you to attend classes.
4. Receive a clearance form for the USC ID card.

Additionally, you can find detailed instructions on online enrollment procedures in ISMIS (Integrated School Management Information System) and the official USC website. You can access ISMIS through the link <https://ismis.usc.edu.ph>.

Fig. 22. Example response with missing chunks

This scenario occurred when the maximum chunk length was configured to 1500 characters or approximately 300-400 tokens, which is way below the average token length per article, yielding a high possibility of losing the proper reference within paragraphs in a chunk. In the figure, the instructions for accessing the enrollment module and choosing the schedules were omitted in the response as the process jumped directly to obtaining the study load. The corrected answer can be found in Fig. 25 where a similar question about the enrollment process is asked and the complete answer is displayed.

Another rare case is shown in Fig. 23 which the model did not answer a question due to the difference in the form the rule was phrased in the question and answer.

It must be completed within the reasonable period set by the faculty member but not to exceed six (6) months following the semester or summer in which the course was taken; otherwise, the grade automatically becomes "NC" (No Credit).

Within how many months can my INC be complied?	I don't know. The provided context does not mention a specific timeframe for complying with an INC (Incomplete) grade.
--	--

Fig. 23. Example response with missed answer

The top part of the Fig. shows the snippet of the article while the bottom right part shows the model's response. This occurred in the grades section concerning the expiration of an incomplete (INC) grade. Instances like this can however be improved through the addition of paraphrased sentences that reiterate the context in a manner that is more similar to how questions regarding that rule is asked.

#### E. Application User Interface

As soon as the parameters were adjusted for the best model performance, the author proceeded to develop the application using the Streamlit framework for Python data science applications along with its deployment to Streamlit Community Cloud. Fig. 24 shows the initial interface of the application resembling the layouts of most social media messaging and chatbot applications.

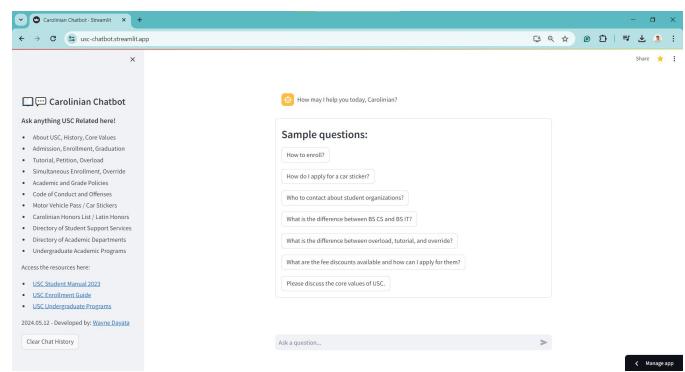


Fig. 24. Initial application user interface

The application initially shows a sidebar listing the related topics and resources, a button to reset chat history, and the main layout displaying buttons of sample questions and a text entry field for users to input their questions.

Fig. 25 shows how the application displays the response from the invoked pipeline, which is shown in two parts: The answer generated by the Llama-3 model and the article names of the chunks retrieved from the document search phase before the model inference phase.

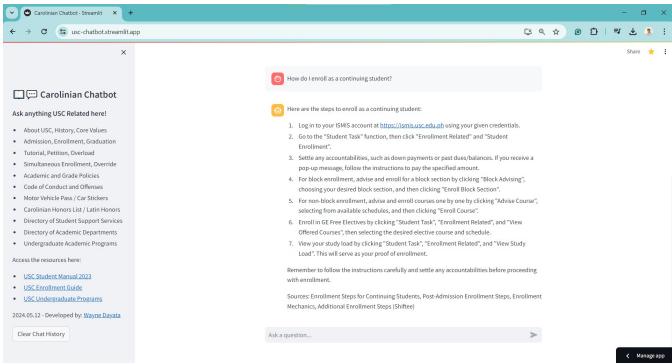


Fig. 25. User interface with LLM responses and sources

As seen, the source containing the enrollment process for freshmen students was included in the retrieval but correctly not stated in the response as the prompt specifically asked for the enrollment guidelines only for continuing students.

#### F. Evaluation

As the app was deployed in Streamlit, several other users managed to access it online and asked questions through the pipeline. Fig. 26 shows the model's responses to some custom questions asked by other users in the system, as logged in the online spreadsheet, showing that the model still performs well on the other questions the author did not use for sampling.

What are the discounts?	There are two fee discounts available for students during enrollment: 1. Family and Sibling Discount: A 5% discount on tuition fees is available for siblings enrolled in USC, except for those enrolled in Montessori, Law, and Graduate Programs. 2. Full Payment Discount: A 5% discount is offered for students who pay their tuition fees in full, with the discount varying depending on the payment method. * 5% discount for Offsite Payment Centers and Onsite Cash payments * 3% discount for Online Credit Card (Visa or Mastercard) payments * 3.5% discount for Online Debit Card payments
What is the family discount?	The Family and Sibling Discount is a 5% discount on tuition fees for siblings enrolled in USC, excluding those enrolled in Montessori, Law, and Graduate Programs.
What are the required documents for Master and Doctoral Program International Applicants?	The required documents for Master and Doctoral Program International Applicants are: 1. International Diploma or its equivalent with term of graduation 2. Physician's Report on a page with at least two areas filled 3. English Proficiency Requirements such as TOEFL or IELTS or at least 60 in TOEFL or as prescribed by the program. Proficiency who graduated from schools that used English as a medium of instruction are required to submit a certificate issued by the head of the institution or its legal representative 4. Proof of Permanent Citizenship for Dual Citizen Applicants such as US/Chinese Passport, Identification Certificate (IC), or Certificate of Registration of Chinese Citizens (CRCC)
What are the requirements for a student to be officially enrolled?	The provided context does not explicitly state the general requirements for a student to be officially enrolled. It provides specific requirements for different types of students, such as international students, cross-enrollees, special students, and graduates from schools abroad. However, it does mention the requirements for newly accepted applicants to prepare before proceeding to enrollment, which include documents such as Form 138, birth certificate, and a 2x2 photo.
Under what circumstances may cross-enrollment be permitted?	Cross-enrollment may be permitted for graduating students with a Deficiency Form, for not more than 6 units during the semester and 3 units during the summer term, inclusive of the total units enrolled in that particular term, and only with the approval of the Department Chair and the School Dean. Additionally, cross-enrollment is not allowed if it is the only remaining course, so as not to prevent the student's residency in the University.
Until when is withdrawal of course(s) allowed?	Withdrawal of course(s) is allowed until mid-term examinations or until the student has incurred unexcused absences (more than 10% of the total required school days).
What might happen to underenrolled classes during the adjustment period?	According to the text, during the adjustment period, underenrolled classes might be dissolved, and students may need to contact their department for resolutions.
What should I do if my class gets dissolved?	If a class gets dissolved during the adjustment period, it may be replaced by an extra class group for a particular course. However, the context does not provide a specific step-by-step guide on what to do if your class gets dissolved. To check if your payment has been cleared, you can follow these steps:
How do I check if my payment has been cleared?	To check if your payment has been cleared, you can follow these steps: 1. Go to "Student Task" 2. Click "Assessment" 3. Click "Student Ledger"
What does INC mean?	This will show you if your payment has been cleared, meaning the university has received bank confirmation. INC means "Incomplete", which is a grade that indicates that the student has an initial passing grade but has not complied with all the academic requirements or failed to take the final examination for the course due to sickness, emergency cases, or accident. It can also mean a failure to submit a major course requirement.

Fig. 26. LLM responses from user questions

However, there are instances like in Fig. 27 where the model did not provide a complete answer as it was not sure whether the rules from the context applied to the situation it provided. Nonetheless, the model still performed well in not providing any assumptions, and it shall be the developer who shall adjust that part of the context specifying the coverage of the rule, in the case of Fig. 27, applying for overload during a summer term.

I do have to apply for overload for summer classes	There is no specific mention of summer classes or whether overload applications are required for them. The context primarily discusses overload policies for regular semesters, and it's unclear if the same rules apply to summer classes. I don't know the answer to this question.
--	---

Fig. 27. Example response from a complex situation

On the other hand, Fig. 28 shows that the model could not interpret "CES Points" and "CES Requirements" as the same entity as their similarity was not mentioned anywhere in the context of the article, resulting in different responses with the lower one ("CES Requirements") being the accurate one.

CES points	I'm not familiar with the term "CES points" in the provided context. CES stands for Community Extension Services, but I couldn't find any information about "CES points" in the provided text. If you could provide more context or clarify what you mean by "CES points," I'll do my best to help.
CES requirements	According to the text, there is no common requirement with regards to CES points that students need to earn prior to graduation. However, some departments require a certain minimum number of CES points that students should acquire to be cleared for graduation. For example, the Department of Computer, Information Sciences, and Mathematics (DCISM) requires 2 CES points from every graduating student, while all the departments in the School of Engineering (SOE) require 60 CES points from every graduating student.

Fig. 28. Variation in responses caused by unidentified similar entities

Likewise, a similar situation is presented in Fig. 29 which is caused by a typographical error by the user in the top row which made the embedding and inference models unable to locate the relevant chunks and answers. This was corrected in the bottom row.

Can you explain more about family privileges?	I apologize, but there is no mention of "family privileges" in the provided context. The topics discussed are related to the Carolinian Honors List, academic policies, code of conduct, and qualifications for the honors list. If you meant to ask about a different topic, please feel free to ask, and I'll do my best to assist you.
	The Family and Sibling Discount is a 5% discount available for siblings enrolled in USC (University of San Carlos). To apply for this discount, a student can follow these steps: 1. Click "Student Task", "Others", and "Apply Same Family Privilege". 2. Click the home icon and indicate the academic period and year. 3. Indicate the ID number of the sibling to be included in the privilege. 4. Click "Submit".
	The application will then be forwarded to VP Finance for approval, and the status can be monitored.
Can you explain more about family privileges?	The discount is 5% on tuition fees, except for siblings enrolled in Montessori, Law, and Graduate Programs.

Fig. 29. Variation in responses caused by typographical errors

In general, although the USC student manual along with the other USC references are informative in its rules, they have not covered certain extreme conditions that could have been answered by the model had they been also specified in the articles, which can most likely be the kinds of questions that student would ask around other students, officers, or staff as they did in the application.

## V. CONCLUSION AND RECOMMENDATIONS

In this project, the author successfully developed a question-answering pipeline covering essential information from the USC Student Manual 2023 and other supplementary articles. A working application was also built and deployed online for other students to use and evaluate. This eliminates the need for students to perform a manual search across the different articles to obtain the answer nor spend time inquiring about their concerns physically at the other offices. However, in critical conditions or complex situations, it is important that the information generated by the model be verified by students in the relevant areas of the student manual, or they can opt to search for the related office contacts as a workaround to inquire about their queries, which is supported in this application due to the directory of contacts being included in the contexts.

More importantly, the question, answer, and source tuples recorded in the online spreadsheet can be used to determine the accuracy of the answers across different contexts and gather insights as to which sources are most frequently utilized by the users in the application. With enough user inputs, the data can be further examined to identify parts of the context that should be clarified for better inference. Moreover, this data can further serve as a recommendation to the university administration to conduct reorientations in the most frequently asked questions and spread awareness in the least utilized offices and services offered.

Finally, the author noted the following recommendations to improve this project further:

- 1) Incorporating models and algorithms that can handle tabular data, which can be applied in the analysis of program prospectus with their pre-requisite and co-requisite courses.
- 2) Gathering data from other university student manuals and strategically storing them to allow comparison of rules across different universities without mixing up the source universities where the rules apply.
- 3) Performing automated tasks such as notifying the offices of the concerns addressed to them as identified by the model for the staff to reply with additional information.
- 4) Experimenting with fine tuning of inference models with smaller resource requirements for better overall efficiency once an adequate amount of question, answer, and source chunk entries have been reached.

## REFERENCES

- [1] M. Caballero, "A brief survey of question answering systems," *International Journal of Artificial Intelligence and Applications*, vol. 12, no. 5, pp. 01–07, Sep. 2021, doi: 10.5121/ijaia.2021.12501.
- [2] M. Sri, "NLP in virtual Assistants," in *Apress eBooks*, 2020, pp. 185–247. doi: 10.1007/978-1-4842-6246-7\_5.
- [3] N. T. K. Le, N. Hadiprodjo, H. El-Alfy, A. Kerimzhanov, and A. Teshebaev, "The Recent Large Language Models in NLP," *2023 22nd International Symposium on Communications and Information Technologies (ISCIT)*, Oct. 2023, doi: 10.1109/ISCIT57293.2023.10376050.
- [4] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. 2024. doi: 10.48550/arXiv.2402.07927.
- [5] Z. Hu et al., "LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models," *arXiv*, Jan. 2023, doi: 10.18653/v1/2023.emnlp-main.319.
- [6] P. S. H. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP tasks," *Neural Information Processing Systems*, vol. 33, pp. 9459–9474, May 2020, [Online]. Available: <https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf>
- [7] V. Pandey, "Should you Prompt, RAG, Tune, or Train? A Guide to Choose the Right Generative AI Approach," *Medium*, Aug. 19, 2023, [Online]. Available: <https://medium.com/@pandey.vikesh/should-you-prompt-rag-tune-or-train-a-guide-to-choose-the-right-generative-ai-approach-5e264043bd7d>
- [8] R. Patil, S. Boit, V. Gudivada, and J. Nandigam, "A survey of text representation and embedding techniques in NLP," *IEEE Access*, vol. 11, pp. 36120–36146, Jan. 2023, doi: 10.1109/access.2023.3266377.
- [9] R. Egger, "Text Representations and Word Embeddings: Vectorizing Textual Data: Interdisciplinary Approaches, Methodologies, and Applications," in *Applied Data science in tourism*, Springer Nature, 2022. doi: 10.1007/978-3-030-88389-8\_16.
- [10] "Introducing Nomic Embed: a truly open embedding model." <https://blog.nomic.ai/posts/nomic-embed-text-v1>
- [11] Z. Nussbaum, J. Morris, B. Duderstadt, and A. Mulyar, "Nomic Embed: Training a Reproducible Long Context Text Embedder," Feb. 2024. doi: 10.48550/arXiv.2402.01613.
- [12] D. Jurafsky and J. H. Martin, "Vector Semantics and Embeddings," in *Speech and Language Processing*, 1st ed., 2024. [Online]. Available: <https://web.stanford.edu/~jurafsky/slp3/6.pdf>
- [13] "Chroma: the AI-native open-source embedding database." <https://www.trychroma.com/>
- [14] X. Amatriain, "Transformer models: an introduction and catalog," *arXiv*, Feb. 2023, doi: 10.48550/arXiv.2302.07730.
- [15] Meta Inc., "Introducing Meta Llama 3: The most capable openly available LLM to date," Meta AI, Apr. 18, 2024. <https://ai.meta.com/blog/meta-llama-3/> (accessed May 15, 2024).
- [16] H. Erz, "The Transformer Architecture: A Visual Guide," *hendrik-erz.de*, Apr. 07, 2024. <https://www.hendrik-erz.de/post/the-transformer-architecture-a-visual-guide-pdf-download> (accessed May 15, 2024).
- [17] P. Khadka, "LLaMA explained! - Pranjal Khadka," *Medium*, Apr. 10, 2024. [Online]. Available: <https://medium.com/@pranjalkhadka/llama-explained-a70e71e706e9>
- [18] University of San Carlos, "University of San Carlos Manual for Undergraduate Students 2023 Edition," Aug. 12, 2023. <https://drive.google.com/file/d/1rFTThhqMrVqMF0k0wMFMOIZuraF4AywYN/view>
- [19] University of San Carlos, "Enrollment Guide for USC Students." <https://enrollmentguide.usc.edu.ph/>
- [20] University of San Carlos, "Enrollment Guide for USC - Contact the Offices." <https://enrollmentguide.usc.edu.ph/contact-the-offices>
- [21] University of San Carlos, "Undergraduate Programs," <https://www.usc.edu.ph/academics/undergraduate-programs>
- [22] "LangChain." <https://www.langchain.com/>
- [23] "Hugging Face – The AI community building the future." <https://huggingface.co/>
- [24] "UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction", *umap 0.5 documentation*. <https://umap-learn.readthedocs.io/en/latest/index.html>
- [25] "Streamlit - A faster way to build and share data apps." <https://streamlit.io/cloud>