SKKU Chatbot for Exchange Students

GeonWoo Bang, Jiae Choi, JungYoon Hwang, and Mikel Larrarte Rodriguez

Sungkyunkwan University Capstone Project 2024

Abstract. The increasing number of exchange students in Korea, driven by the global popularity of Korean culture, highlights the need for better access to university-related information. Sungkyunkwan University (SKKU) is a popular destination for these students, but finding relevant information is challenging due to language barriers and fragmented resources. Existing chatbot solutions are limited by monolingual capabilities and keyword-based search methods, restricting their effectiveness. To address these issues, this paper proposes a RAG (Retrieval-Augmented Generation) based chatbot model that efficiently delivers information to exchange students. The chatbot retrieves relevant documents from a pre-constructed database of web-scraped content and generates responses tailored to the user's query. Additionally, the chatbot can provide full scripts and links to the retrieved documents upon request, mitigating potential inaccuracies.

Keywords: Chatbot · RAG · LLM

1 Introduction

Number of exchange students is increasing every year largely due to the growing global popularity of Korean culture such as K-pop, and K-drama. Among many universities in Korea, Sungkyunkwan University(SKKU) is one the most visited university by exchange students. [1]

However, it is challenging to find vital information about the university for the exchange students, primarily due to the language barrier, and the fact that relevant information is scattered apart across multiple platforms. [2]

Previous works on delivering relevant information by a chatbot[3][4] are limited by their monolingual nature and keyword-based search methods, which reduces the diversity of possible queries and fails to capture context dependency.

To address these challenges of the exchange students and the limitations of the previous works, we propose a RAG based chatbot model to deliver information efficiently and user-friendly. When a user query is given, the RAG model retrieves relevant documents from a pre-constructed database in which scraped documents from the web are stored. Then the model generates responses based on the user query and the retrieved documents. Finally, the user is informed by the model's response. If requested by the user, the model can also provide the full script of retrieved documents and their links, to reduce some possible hallucinations.

We divided our project into several sub tasks as follows: Scraping, constructing database, implementing the RAG model(retriever and generator), implementing back-end, implementing front-end, and evaluating the model.

The rest of this proposal is divided into four sections:

Motivation detailed problem statement of our project.

Background/Related works details about the previous works and required background knowledge in order to implement our method.

Method detailed explanation and implementation planning about our method.

Planning & role assignment Scheduling of the sub tasks and assignments.

2 Motivation

Exchange students at Sungkyunkwan University (SKKU) often struggle to access vital information about the university's academic and administrative systems due to language barriers and the unfamiliar environment. Moreover, non-native speakers find it hard to navigate as essential information such as course registration, campus maps, and extracurricular activities are frequently dispersed across different platforms, often only available in Korean. These challenges hinder students' ability to settle into campus life smoothly and fully benefit from their time at SKKU.

With the rise of AI technologies nowadays, one of the most efficient and user-friendly ways to deliver information is through a chatbot. It allows users to receive instant responses to their queries while interacting in a familiar and concise way. A chatbot can act as a centralized tool for providing answers to frequently asked questions, such as academic and social inquiries about the campus. For exchange students, a chatbot offers the convenience of quick and easy access to information without the need to navigate multiple university websites or communication systems in a foreign language.

Despite the availability of chatbots at some institutions, most existing solutions have limited scope and functionality. Typically, these chatbots are either monolingual, designed primarily for native students, or rely on simple keyword-based matching systems, which restrict their ability to handle diverse and context-dependent queries. Such limitations make the need for a more advanced, multilingual solution more evident. The chatbot we propose will be based on Retrieval-Augmented Generation (RAG), enabling it to not only respond to queries in multiple languages but also generate accurate, contextually relevant responses based on up-to-date information obtained from websites related to SKKU. By addressing the needs of SKKU's student diversity, this system will significantly improve exchange students' ability to access and understand critical information efficiently.

3 Background/Related works

3.1 Background

Scraping Scraping is a method of automatically extracting needed data from a specific website. It involves analyzing the website's HTML to gather the required information. In LangChain, we can use AsyncHtmlLoader or AsyncChromium-Loader to convert a URL into HTML, and then utilize HTML2Text or Beautiful Soup to convert the HTML into formatted text.[5]

PDF to Text To use the content of PDF files with an LLM, converting the PDF into text is necessary. PDFs can be categorized into image-based PDFs and text-based PDFs. When dealing with image-based PDFs, people typically use Optical Character Recognition (OCR) to extract text from the images, whereas text can be directly extracted from text-based PDFs without OCR. In LangChain, libraries such as 'pypdf' and 'langchain-unstructured' allow PDFs to be used as input data for LLM models. 'pypdf' does not provide OCR, so it cannot extract text from images in PDFs. On the other hand, 'langchain-unstructured' offers OCR, enabling the extraction of text from images as well.[6]

RAG Retrieval-Augmented Generation (RAG) is a technique that enhances natural language processing (NLP) models by integrating external knowledge retrieval into the generation process.[7] Traditional large language models rely solely on their training data, which can result in outdated or inaccurate information, especially regarding events that occurred after their training period.

While fine-tuning pre-trained language models on specific datasets can improve performance for certain tasks, it has several limitations:

Static Knowledge Base Fine-tuned models retain only the information present during training and cannot dynamically incorporate new data.

Computational Cost Updating the model to include new information requires retraining, which demands significant computational resources.

Limited Generalization Overfitting to the fine-tuning dataset may reduce the model's ability to generalize to unseen data or broader topics.

RAG overcomes these limitations by integrating an external retrieval mechanism into the generation process. The principal components of RAG are as follows:

Information Retrieval Module Searches for relevant documents or data from external databases based on the query. Techniques like Dense Passage Retrieval (DPR) [8] or Facebook AI Similarity Search (FAISS) [9] are used to compute the similarity between the query and documents in a high-dimensional vector space.

Generation Model Generates accurate and reliable text by using the retrieved information as additional context. Transformer-based language models are commonly used, processing the input query along with the retrieved documents to produce responses.

The operation process of RAG is as follows:

Step 1: Information Retrieval For the input query, the information retrieval module searches for relevant documents from the external knowledge base. Using DPR or FAISS, it calculates the similarity between the query and documents and selects the top n relevant documents.

Step 2: Text Generation The generation model takes the input query along with the retrieved documents to generate an accurate response that is contextually appropriate. In this process, the model reflects the latest and accurate content based on the retrieved information.

Table 1. pros and cons of RAG compared to traditional fine-tuning methods

Aspect	**Fine-Tuning**	**Retrieval-Augmented Generation (RAG)**
Knowledge Update	Requires retraining to incorporate new information; static knowledge base	Dynamically integrates new information through external retrieval methods, like FAISS
Computational Cost	High; retraining is time-consuming and resource-intensive	Low; avoids retraining by utilizing efficient retrieval mechanisms
Response Accuracy	May generate outdated or incorrect information due to reliance on static data	Provides accurate and up-to-date information by grounding responses in retrieved data
Generalization	Limited; may overfit to specific datasets and struggle with unseen queries	Better generalization; handles a wide range of topics and queries through dynamic retrieval
Scalability	Less scalable; significant computational resources needed to manage large datasets	Highly scalable; efficiently manages large external corpora using tools like FAISS
Implementation Complexity	Simple; applies additional training to existing models	Complex; requires integration of retrieval and generation modules, but offers flexibility and long-term efficiency

Table. 1 summarizes the advantages and disadvantages of RAG compared to traditional fine-tuning methods:

By integrating external knowledge retrieval, RAG overcomes the limitations of fine-tuning and provides an efficient solution for generating accurate and contextually appropriate text with no retraining. This is particularly useful in applications requiring access to the latest information, such as open-domain question answering, knowledge-intensive tasks, and conversational agents.

3.2 Related works

Kingobot Kingobot is a school information chatbot from SKKU. It provides convenience to students by gathering scattered school information in one place. Students can use Kingobot in KINGO-M (SKKU's official mobile service application) and school web page (https://www.skku.edu). However, exchange students may face some difficulties using Kingobot.

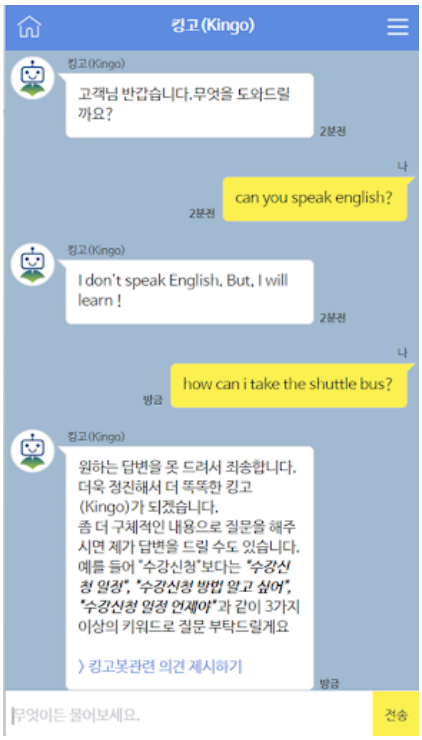

Fig. 1. Kingobot usage screen(asking in English)

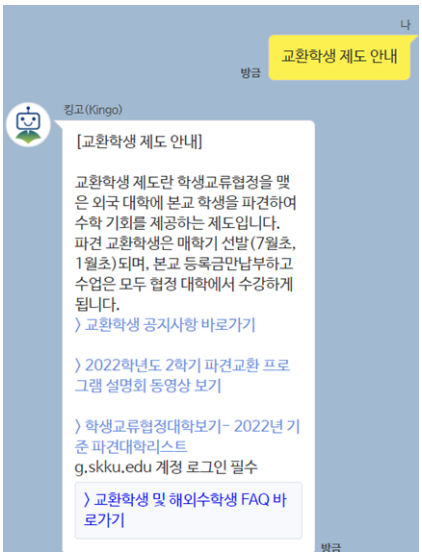

Fig. 2. Kingobot usage screen (little information for international students)

First, as shown in Fig. 1, it doesn't support English or any other foreign languages. To use Kingobot, exchange students should be fluent in Korean. Second, there are slightly strict rules when asking questions, in instance, "ask with up to 3 keywords ". Rules for asking questions make it difficult for users to obtain information naturally and comfortably. Finally, as shown in Fig. 2, most of the information provided by Kingobot is aimed at general students, with very little available for exchange students. We need a chatbot that specifically includes information for exchange students as well.

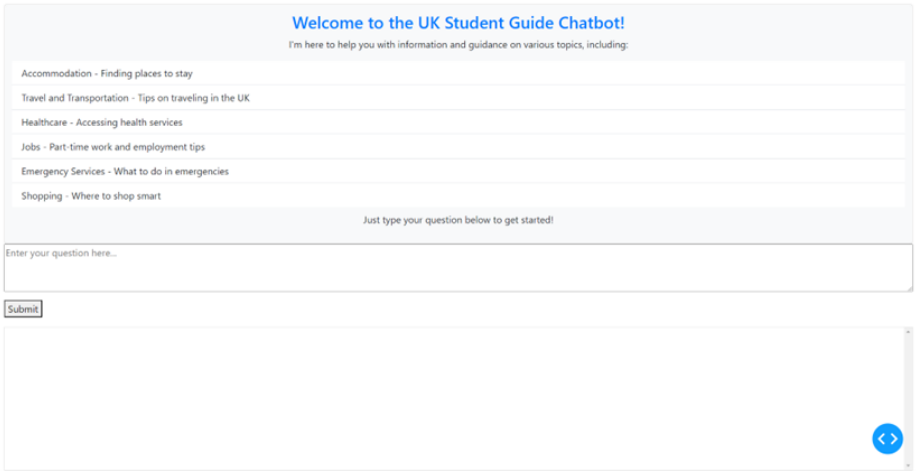

Fig. 3. International student chatbot for UK students

International student chatbot for UK students This is a chatbot service for international students in the UK, which provides answers based on keywords. The chatbot is designed to respond only when a preprocessed question keyword matches the chatbot's intent keywords. As a result, the range of acceptable questions is limited, and users may not receive the desired answer depending on how the question is phrased. For example, if the chatbot's intent keywords only include 'transportation' and a user asks about the 'shuttle bus', the chatbot response may be "can not understand the question", even though the purpose of the question is almost the same. In addition, this chatbot covers a wide range of topics, making it difficult to provide detailed information. Since each school has unique information needs, a customized chatbot for each school would be more effective.

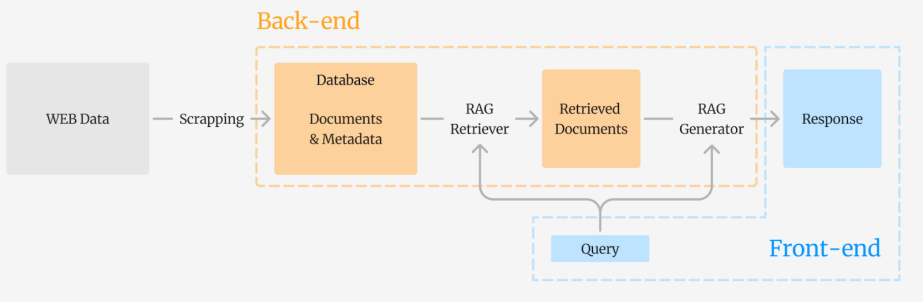

Fig. 4. Overview of our proposed method

4 Method

4.1 Overview

Fig. 4 is an overview of our proposed method. In a nutshell, it is composed of three sections - scraping, RAG, and front-end.

Scraping Scrap the web pages with information about SKKU and construct a database for the RAG model to retrieve relevant documents from.

RAG With the constructed database, the model returns a response given a query from the front-end.

Front-end Web application that outputs RAG's response given user's query. When requested, the front-end also receives full documents that RAG retrieved, and the link of the documents.

The following sections are also divided based on these three parts, with one additional part about an interface between the front-end and the back-end.

4.2 Scraping

Dataset To develop our chatbot for exchange students in SKKU, we will use these datasets.

- Office of International Student Services, Sungkyunkwan University Webpage data [10] (introduction, immigration guided, campus life, academics, etc)
- Today's School restaurant menu [11]
- 2024 SKKU club guide book [12]
- International Student Handbook [13]
- (maybe) Everytime data: a school community app

Scraping We will get general school information for exchange students and school restaurant menus by scraping Office of International Student Services, welfare tab of Sungkyunkwan University Webpage. We will use LangChain library(AsyncHtmlLoader, AsyncChromiumLoader, HTML2Text, Beautiful Soup) to scrape these webpages.

If we're able to scrape data from Everytime, we can gather lecture reviews left by SKKU students. However, since the Everytime account used for scraping could be suspended, it's important to create a scraper with sufficient sleep time.

PDF to Text We will also gather general school information for exchange students and details about SKKU's clubs from the International Student Handbook and 2024 SKKU Club Guidebook. As many sections are in image format, we'll use the langchain-unstructured library to load PDF files into the LangChain document format.

4.3 RAG

We will gather Korean and English data by crawling publicly available information from various web sources using tools like the LangChain library (AsyncHtmlLoader, AsyncChromiumLoader, HTML2Text, Beautiful Soup) and official pdfs. The crawled data will include textual information related to topics relevant to our RAG (Retrieval-Augmented Generation) pipeline.

Once the data is collected, we will preprocess it to extract and encode the relevant textual content into Korean-English embeddings using pretrained models uploaded on Hugging Face. These embeddings will then be stored in an external vector database, such as Pinecone, which allows for scalable and high-performance storage of vectorized information. For efficient retrieval, we will employ FAISS (Facebook AI Similarity Search), enabling us to perform fast and accurate similarity searches across the Korean-English embeddings.

The retriever will query this vector database to fetch the most relevant contexts based on a user query. The retrieved contexts will be passed into the generator prompt part of our system, where we plan to use Llama 3.1-instruct or a Korean-English bilingual pretrained model (such as a Llama-based model). This model will take the retrieved content as context to generate coherent, contextually accurate responses or documents in the target language.

4.4 Front-end

Responsive Web We will develop a user interface with responsive web. Responsive web has the advantage of being accessible on all devices and browsers, and it is easy to develop and test within a short period. [14] We determined that responsive web is the most suitable for developing an interface for users to access our chatbot on any device within 10 weeks. Additionally, responsive web ensures that users are always using the latest version, without users needing to manually

update the service. This is particularly beneficial for chatbot services, where reflecting the latest data quickly is important. Furthermore, since responsive web is URL-based, if this chatbot goes into actual use, it can be easily integrated into existing systems, such as linking it to the school website like KingoBot.

Tools(React.js, Typescript We can easily implement a responsive web design using react-responsive in React.js. By using TypeScript, we can catch type errors at compile time, improving code safety.

4.5 Back-end

When designing the back-end, our team prioritizes developing a reliable and structured solution, always keeping accuracy and efficiency in mind. It should be responsible for handling user queries, retrieving relevant information, generating appropriate responses, and ensuring that these interactions are seamless across multiple languages.

First of all, we have to specify the programming language in which we will develop the entire infrastructure. We chose Python for the backend due to its wide range of libraries suited for natural language processing (NLP), API development, and integration with machine learning models (Hugging Face). Python's flexibility makes it the optimal choice for this project, particularly when working with complex tasks like document retrieval and generation.

Regarding the libraries to use in our project, we are considering several options, such as FastAPI, for building a lightweight and scalable REST API, LangChain, to handle document retrieval and loading, and Hugging Face, to implement the RAG model.

FastAPI allows us to build a robust REST API that can handle multiple concurrent requests efficiently, a crucial feature for chatbots handling various users at the same time. One of the key benefits is that it is one of the fastest Python frameworks available. Moreover, its usage is not that complex, and it grants a major advantage as it provides automatic API documentation, so we do not have to worry about that aspect of the developing process, allowing us to build the API within the timing constraints. The API will serve as the intermediary between the frontend chatbot interface and the backend processes, handling requests from users, retrieving data, and generating responses.

Instead of using a specific tool for document retrieval, we leverage LangChain's built-in retrievers. These retrievers pull relevant documents from a variety of sources, including PDFs, websites, and databases, making LangChain a flexible solution. Basically, we use LangChain's vector database capabilities (e.g., FAISS) to store and retrieve documents based on the semantic similarity of the user's query. When a query is made, LangChain searches through the stored documents, retrieves the most relevant ones, and passes them to the response generation component. In summary, it simplifies retrieval by allowing us to focus on how the documents are used, rather than how they are stored, offering flexibility to swap in other retrieval systems if necessary.

LangChain also gives the opportunity to build the generator part of the RAG system. For that reason, for generating responses, we are contemplating the option of using LangChain's implementation of Retrieval-Augmented Generation. This approach retrieves relevant documents and passes them to a generative model to create a detailed, contextually accurate response. In addition, the Hugging Face package langchain-huggingface integrates seamlessly with LangChain, providing an efficient and effective way to utilize Hugging Face models within the LangChain ecosystem.

5 Planning/Role assignment

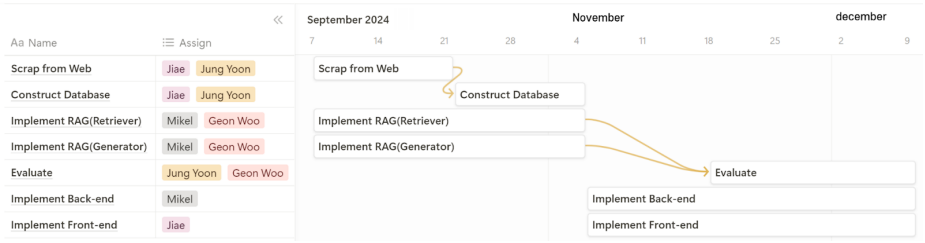

Fig. 5. Gantt chart of our project

We divided our project into several sub tasks.

Scrap from web Find web pages with information about SKKU and scrap them.

Construct database With the scraped data, construct a database for RAG to retrieve relevant documents from.

Implement RAG retriever Exploring optimal vector similarity calculation methods and building a retriever for efficient information retrieval.

Implement RAG generator Building a chatbot using an LLM model based on the context provided by the retriever.

Evaluate Construct an evaluation dataset and evaluate the implemented RAG retriever and generator separately.

Implement Back-end Implement the inference code for RAG to communicate with the front-end.

Implement Front-end Implement web application that outputs RAG's response given user's query. The sub tasks are scheduled and assigned using the gantt chart(Fig. 5.). However, the planning is open to changes, as we adopted the agile methodology.

References

1. The JoonAng, https://www.joongang.co.kr/article/25160711, last accessed 2024/10/2.
2. SKKU news, https://www.skkuw.com/news/articleView.html?idxno=24288, last accessed 2024/10/2
3. Kingobot, https://kingo.skku.edu/chat, last accessed 2024/10/2
4. BalaElangovan Github, https://github.com/BalaElangovan/International-student_Chatbot?tab=readme-ov-file, last accessed 2024/10/2
5. LangChain Web Scraping, https://python.langchain.com/v0.1/docs/use_cases/web_scraping/, last accessed 2024/10/2
6. LangChain How to Load PDFs, https://python.langchain.com/docs/how_to/document_loader_pdf/, last accessed 2024/10/1
7. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In: Advances in Neural Information Processing Systems 33 (NeurIPS 2020), pp. 9459–9474 (2020). https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
8. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.-t.: Dense Passage Retrieval for Open-Domain Question Answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 6769–6781 (2020). https://doi.org/10.18653/v1/2020.emnlp-main.550
9. Johnson, J., Douze, M., Jégou, H.: Billion-scale similarity search with GPUs. IEEE Transactions on Big Data **7**(3), 535–547 (2021). https://doi.org/10.1109/TBDATA.2019.2921572
10. SKKU Office of International Student Services, https://oiss.skku.edu/oiss/index.do, last accessed 2024/10/1
11. SKKU Welfare Services, https://www.skku.edu/skku/campus/support/welfare_11.do, last accessed 2024/10/1
12. AKDONG, 2024 SKKU club guide book, 2024. Available: https://drive.google.com/file/d/1S_aVDkmrwaJaT1sWVO67b8mubEIrHLOU/view?pli=1
13. International Student Handbook, https://ibook.skku.edu/Viewer/9WLAOPP0LYA9, last accessed 2024/10/1
14. WEZOM, https://wezom.com/blog/mobile-app-vs-responsive-website-which-is-the-better, last accessed 2024/10/1