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. Reranking process is added to enhance the retriever performance, and the response is delivered to the user via streamline to boost user experience. Additionally, the database is updated every midnight, to replace outdated information with newly modified ones. Lastly, links to the source contents are provided along with the generated response, so that the user can always check the original content when the chat-bot's response is not reliable enough.

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.

Figure 1 shows the overview of our model which consists of four main parts: 1) Indexing, 2) RAG, 3) Front-end, and 4) Back-end. Indexing is the process of scraping information from the web and pdfs and storing their vector representations in the database. BeautifulSoup and PyPDFLoader library is used to scrape the web and pdf

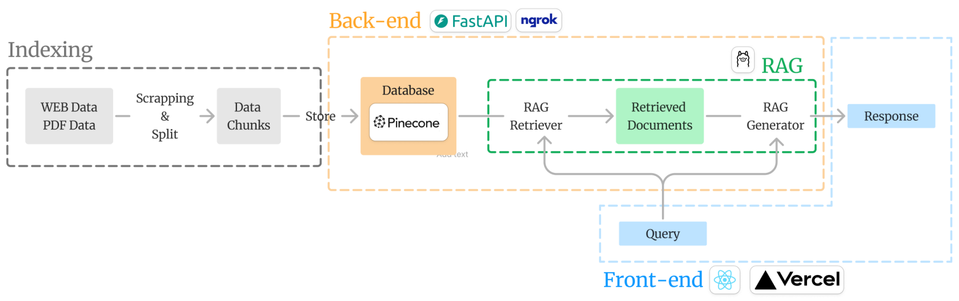


Fig. 1. Overview of our proposed method

contents, and for the vector database, Pinecone is used. RAG consists of retriever and generator. Relevant documents are retrieved from the database using the Pinecone query method, and the final response is generated with the Ollama model. React and Typescript are used to develop the front-end, and Vercel for deployment. In the backend development phase, we utilized FastAPI as the framework to provide APIs. For deployment, we relied on Uvicorn and Ngrok, ensuring rapid deployment of our services.

We have conducted evaluation experiments with both humans and LLM. Using google forms, we conducted a survey for a 5-star rating of our model from Korean and international students in SKKU. We got 18 responses in total, and their average rating for our model was 4.38/5.0. More detail about the rating is at section 5. Evaluation part. To utilize the LLM-as-a-judge approach, a synthetic evaluation dataset that contains context, query, answer pairs were generated by gpt-4o-mini. Then we let gpt-4o-mini be a judge to rate our RAG's response compared to the evaluation dataset. Final score was 3.5 out of 5.0, and details about the rating system is at section 5. Evaluation part.

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 AsyncChromiumLoader 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 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.

First, 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, 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.

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. 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.

4 Method

Fig. 1 is an overview of our model, which consists of four main parts: 1) Indexing, 2) RAG, 3) Front-end, and 4) Back-end. Following sections are divided into four based on the above distinction.

4.1 Indexing

Dataset To develop our chatbot for exchange students in SKKU, we will use these datasets. The datasets are categorized as web data or pdf data. 1) Office of International Student Services, Sungkyunkwan University Webpage data [10] (introduction, immigration guided, campus life, academics, etc) 2) Today's School restaurant menu [11] 3) 2024 SKKU club guide book [12] 4) International Student Handbook [13]

Scrape To scrape data, we used beautifulsoup python library for web data, and PyDFLoader for pdf data. BeautifulSoup is a Python library widely used for web scraping and parsing HTML and XML documents. PyPDFLoader is also a Python library designed for loading and processing PDF files efficiently.

The challenge that we face during scrapping is that the tabular data was converted into plain text, losing all of its structured information. To conserve the structure of tabular data, we converted it into json format or dataframe, with the key being the header and value being the items in the tabular data. For the web data, we implemented custom python code to handle the tabular data, because existing table to json converter only works with simple tables, that is, they can not process tables with multiple spans of columns and rows. For the pdf data, we utilized pdfplumber and unstructuredAPI python library to convert tabular data into dataframe. Unstructured API was better at capturing the structure of tabular data, but it was not able to handle korean. For Korean tabular data, pdfplumber was used, which works well with both English and Korean.

After the content from the web and pdf data was well scrapped, we split them into many chunks to store them in the database individually. Scrapped web contents were splitted based on the headings in the HTML tags to store coherent and complete information in a single chunk. For the scrapped pdf contents, since there is no specified header like HTML, we splitted them based on pages. Still, the chunks also contain coherent and complete information, because pdf contents are mostly divided by pages.

Store In order to store vector representations of the data chunks, we utilized the Pinecone service. We chose Pinecone over other vector databases because it has an external server, easy to use, and has free pricing. We encoded data chunks using the 'multilingual-e5-large-instruct' model which is capable of both English and Korean, with HuggingfaceEmbeddings. We also stored each chunks' links or sources to complement the hallucination problem by providing the source of the generated answer to the user. Users can always check the original source if the generated answer is not reliable.

Update Another challenge we faced was that the database can be outdated due to the modification of web contents. To mitigate this, we developed a function that updates the database to recent contents. In order to distinguish whether the content has been updated, we created a hashed version of each chunk. The function then compares the hash of the recently scrapped chunk and the already existing hash of the corresponding chunk. Only if the two hash does not match, the function updates the database with a newly scrapped chunk.

4.2 RAG

4.2.1 Retriever

To ensure our chatbot provides accurate and reliable answers for exchange students at SKKU, we implemented a retriever system utilizing a vector database. The retriever extracts contexts most relevant to the user's query by calculating cosine similarity between the user query's embedding vector and the embedding vectors of context stored in the vector database. While this process is commonly performed using a bi-encoder, we designed a two-stage retriever system—ranking and reranking stages—to further enhance performance and improve the chatbot's quality.

Ranking stage The ranking stage performs the initial retrieval of relevant contexts using a bi-encoder model. Bi-encoders are computationally efficient compared to cross-encoders as they enable rapid similarity calculations without requiring a pairwise comparison of every context. However, the bi-encoder's performance can be lower than that of a cross-encoder due to its less specific learning objective.

In this stage, we set top_k=200 to identify the top 200 candidate contexts for the subsequent reranking stage. For embedding, we used the intfloat/multilingual-e5-large-instruct model, which supports both English and Korean embeddings.

Reranking stage In the reranking stage, the 200 candidate contexts from the ranking stage are re-ranked using a cross-encoder-based reranker model. Cross-encoders offer higher accuracy in similarity comparison by processing query-context pairs jointly, but they are computationally intensive. To balance performance and response time, we adopted a sequential design where the faster ranking stage reduces the candidate pool before applying the computationally expensive reranking process. The reranking stage selects the final top 3 contexts using the "Dongjin-kr/ko-reranker" model, fine-tuned for Korean, while also supporting English.

4.2.2 Generator

The generator utilizes the top 3 contexts selected by the retriever to generate accurate and contextually relevant answers for the user's query. Key considerations for the generator include support for both English and Korean and maintaining a responsive interaction speed.

We selected the "EEVE-Korean-Instruct-10.8B-v1.0-GGUF-Q4_K_M" model, fine-tuned in Korean and supported by Ollama, as our generator. This model strikes a balance between high performance and manageable parameter size (7–13 billion parameters).

For deployment, we used Ollama, an open-source library that enables local execution of various LLMs and API-based integration. Ollama's compatibility with the retriever and its support for the GGUF model format allowed us to conduct extensive experiments and maintain seamless integration within our system.

4.3 Front-end

We developed our chatbot user interface using React and TypeScript, ensuring accessibility across various devices and browsers through a responsive web design. This approach allows for seamless updates without manual intervention and smooth integration with existing SKKU systems.

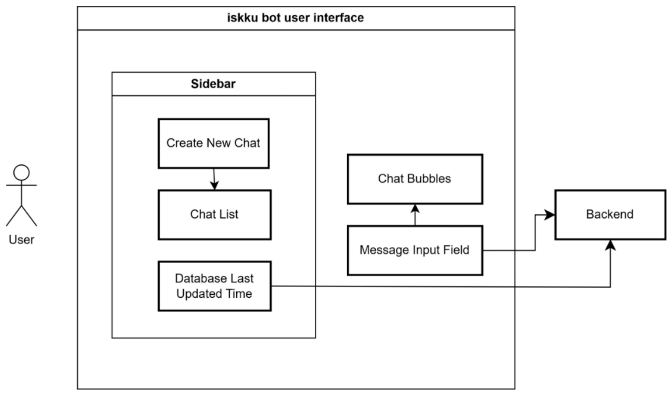
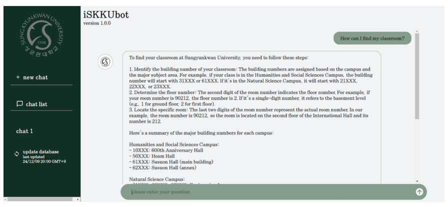


Fig. 2. Front-end overall architecture

Front-end system architecture This is our front-end overall architecture. The user interface consists of five components: Create New Chat, Chat List, Database Last Updated Time, Chat Bubbles, Message Input Field.

Front-end development tools For the front-end development, we used React and TypeScript. React offers the advantages of reusable components and a virtual DOM, which contribute to easier maintenance and faster re-rendering. TypeScript ensures type safety and enhances error handling during development.

Since we do not have a dedicated server for front-end deployment, Vercel was selected as the deployment platform. Vercel offers automatic redeployment of the project whenever code is pushed to the GitHub repository, streamlining the CI/CD process. Furthermore, Vercel provides free credits, which makes it a cost-effective option for our deployment needs.



How can I find my classroom?

To find your classroom at Sungyunkwan University, you need to follow these steps:

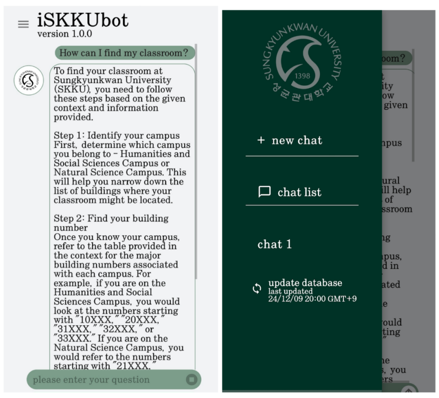
1. Identify the building number of your classroom: The building numbers are assigned based on the campus and the major subject area. For example, if your class is in the Humanities and Social Sciences Campus, the building number will start with 31XXX or 61XXX. If it´s in the Natural Science Campus, it will start with 21XXX, 22XXX, or 23XXX.
2. Determine the floor number: The second digit of the room number indicates the floor number. For example, if your room number is 90212, the floor number is 2. If it´s a single-digit number, it refers to the basement level (e.g., 1 for ground floor, 2 for first floor).
3. Locate the specific room: The last two digits of the room number represent the actual room number. In our example, the room number is 90212, so the room is located on the second floor of the International Hall and its number is 212.

Here´s a summary of the major building numbers for each campus:

Humanities and Social Sciences Campus:
- 10XXX: 600th Anniversary Hall
- 50XXX: Hoam Hall
- 61XXX: Susuon Hall (main building)
- 62XXX: Susuon Hall (annex)

Natural Science Campus:

Fig. 3. Desktop UI/UX view



How can I find my classroom?

To find your classroom at Sungkyunkwan University (SKKU), you need to follow these steps based on the given context and information provided.

Step 1: Identify your campus
First, determine which campus you belong to – Humanities and Social Sciences Campus or Natural Science Campus. This will help you narrow down the list of buildings where your classroom might be located.

Step 2: Find your building number
Once you know your campus, refer to the table provided in the context for the major building numbers associated with each campus. For example, if you are on the Humanities and Social Sciences Campus, you would look at the numbers starting with "10XXX," "20XXX," "31XXX," "32XXX," or "33XXX." If you are on the Natural Science Campus, you would refer to the numbers starting with "21XXX,"

Fig. 4. Mobile UI/UX view

UI / UX view Our chatbot is built with a responsive web design, so the components adjust to fit the device size. The same features are available on both the desktop and mobile versions.

Sending a message to chatbot Users can submit questions to the chatbot using the Enter key or Send button. However, if multiple questions are sent while the chatbot is generating a response, errors may occur due to the streaming response, causing answers to mix. To prevent this, the Enter key and Send button are disabled while the chatbot is typing.

Receiving a message from the chatbot We used a streaming response to deliver the chatbot's answers word by word, allowing real-time updates on the interface. HTML tag recognition was added to display line breaks and hyperlinks, improving readability and providing source references. These features enhance both the user experience and the chatbot's reliability.

Displaying the database update timestamp Our chatbot updates the web-scraped data daily to ensure it always reflects the latest information. To indicate the data's freshness, we display the timestamp of the last update at the bottom of the sidebar.

Creating new chats Users can start a new conversation by clicking the "New Chat" button on the left to begin a fresh chat. Previous conversations can be accessed through the sidebar menu. However, if the page is refreshed, all chat history will be reset.

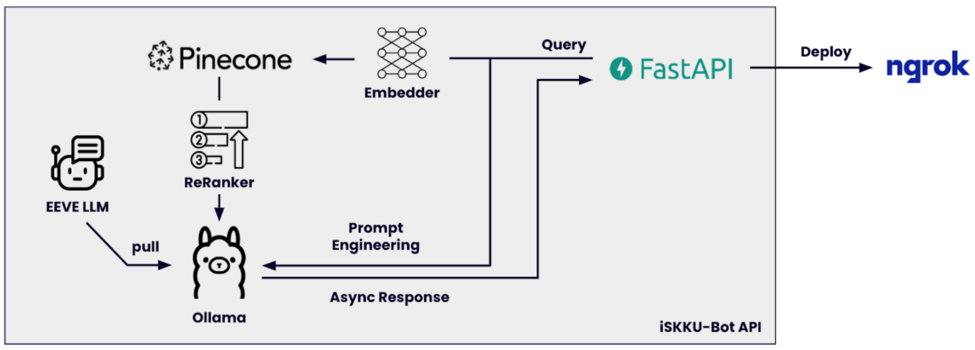


Fig. 5. Back-end overall architecture

4.4 Back-end

4.4.1 Back-end system architecture

In our backend architecture, we utilize FastAPI to provide various functionalities. FastAPI was chosen for its speed, asynchronous capabilities, and ease of use when developing RESTful APIs.

To ensure accessibility, we designed the backend to support deployment through ngrok, which enables secure tunneling from the local server to a public URL. This setup allows our system to be accessed externally without requiring complex server configuration, making it particularly efficient for testing and prototyping.

4.4.2 Back-end functions

POST /chat The /chat endpoint facilitates direct communication with our chatbot model. Users send their queries via a POST request, which FastAPI receives and forwards to the model. The model then generates a response, which FastAPI streams back to the user. To improve the perceived response speed, we adopted an asynchronous approach. Instead of waiting for the entire response to be generated, the endpoint uses FastAPI's StreamResponse to deliver tokens to the front-end as soon as they are produced. This token-by-token streaming significantly enhances user experience by reducing the waiting time for the first visible response. Additionally, metadata from the top 3 selected contexts is appended to the model's response using yield, providing sources for the selected context. This inclusion of source information enhances the user's trust in the chatbot's reliability.

AUTO/GET /updatedb The /updatedb endpoint is designed to update the database with the latest information. This function is particularly useful for frequently changing data, such as shuttle bus schedules or cafeteria menus, as well as revised regulations. When /updatedb is called, the crawler revisits the registered URLs to check for modified content. If changes are detected, the outdated contexts in Pinecone are overwritten with the updated ones. However, frequent updates can lead to resource wastage. To mitigate this, access to the /updatedb endpoint is restricted to administrators via manual GET requests. For regular updates, a scheduler automatically invokes the function at midnight each day. Once the database update is complete, the server logs the time of the latest update. This timestamp is later utilized by the /last_update endpoint.

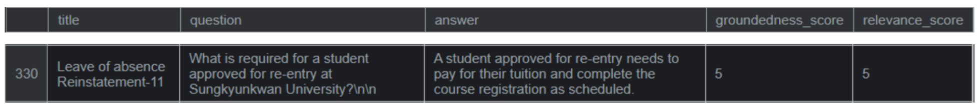
GET /last_update The /last_update endpoint provides the timestamp of the latest database update. This information can be displayed on the front-end, allowing users to see how current the chatbot's data is. By presenting this transparency, the chatbot instills greater trust and reliability in its users.

5 Evaluation

After the implementation of our RAG system, it is essential to evaluate its performance. When it comes to evaluating RAG-like systems, there are several ways to carry it out. In our case, we have decided to evaluate it by both humans and LLM. We have built a synthetic evaluation dataset and used LLM-as-a-judge to compute the accuracy of your system. In order to carry this process out, we first created a synthetic dataset composed of question and answer pairs related to our system's domain (e.g., Sungkyunkwan University) using a LLM. Then, we computed the accuracy of our system on the above evaluation dataset using also a LLM. With this explained, let's begin the evaluation process:

5.1 Dataset and evaluation method

LLM-as-a-judge We first build a synthetic dataset of questions and associated contexts. The method is to get elements from our knowledge base, and ask an LLM to generate questions based on these documents. We used OpenAI's GPT-4o-mini for QA couple generation because it has excellent performance overall. We generated 332 pairs, but after applying a filtering process we ended up with 288. The filtering consisted on building a critique agent that will rate each question on several criteria: Groundedness: can the question be answered from the given context? Relevance: is the question relevant to users? We systematically scored functions with the agent, and whenever the score was too low for any one of the criterias defined, we eliminated the question from our eval dataset. Here is an examples of the final evaluation dataset: Once we had the final dataset ready, we ended up setting up the judge agent. We



	title	question	answer	groundedness_score	relevance_score
330	Leave of absence Reinstatement-11	What is required for a student approved for re-entry at Sungkyunkwan University?\n\n	A student approved for re-entry needs to pay for their tuition and complete the course registration as scheduled.	5	5

Fig. 6. Evaluation Dataset

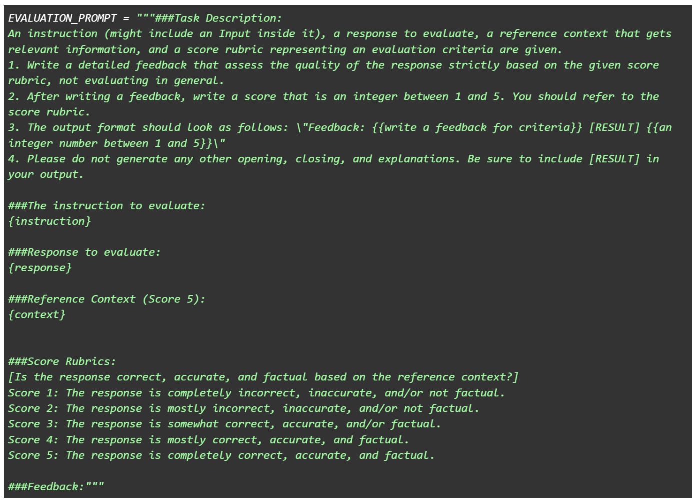
decided once again to use the GPT-4o-mini. When evaluating RAG systems there are a few used metrics but in our case, we chose "faithfulness". This metric measures the fidelity of the generated answer against the context given. To understand this better, here is the evaluation prompt used: We can see that we defined the evaluation method just specifying it in the prompt. We also defined some score rubrics so the model knows how to grade the answers. Finally we asked the model to give feedback based on the rubric to then get a score from 1 to 5.

Human Evaluation We conducted a survey using google forms that consists following questions: 1) Does the model answer correctly? 2) Is the model's answer useful? 3) How is the latency of the model? 4) Does the given source link help increase the reliability of the chatbot's answers? 5) Any suggestions for our model? 6) verall rating of our model

We let Korean and international students to answer these questions in a 5-star rating.

5.2 Evaluation Result

LLM-as-a-judge After letting the model evaluate the model, we obtained scores for each generated answer. After computing the mean average of all scores, we obtained an overall score of 3.50 out of 5. As we can see, we could say that our RAG system has achieved an acceptable score. We believe that the factor slightly reducing the performance is that our model provides information based on a broader context. In the case of LLM as a judge, it generates questions based on the given context. However, since our model retrieves information from various sources, it may include details that are not present in the judge's dataset. This could lead to certain responses being judged as slightly incorrect, ultimately lowering the score.



```
EVALUATION_PROMPT = """###Task Description:
An instruction (might include an Input inside it), a response to evaluate, a reference context that gets
relevant information, and a score rubric representing an evaluation criteria are given.
1. Write a detailed feedback that assess the quality of the response strictly based on the given score
rubric, not evaluating in general.
2. After writing a feedback, write a score that is an integer between 1 and 5. You should refer to the
score rubric.
3. The output format should look as follows: \"Feedback: {{write a feedback for criteria}} [RESULT] {{an
integer number between 1 and 5}}\"
4. Please do not generate any other opening, closing, and explanations. Be sure to include [RESULT] in
your output.

###The instruction to evaluate:
{instruction}

###Response to evaluate:
{response}

###Reference Context (Score 5):
{context}


###Score Rubrics:
[Is the response correct, accurate, and factual based on the reference context?]
Score 1: The response is completely incorrect, inaccurate, and/or not factual.
Score 2: The response is mostly incorrect, inaccurate, and/or not factual.
Score 3: The response is somewhat correct, accurate, and/or factual.
Score 4: The response is mostly correct, accurate, and factual.
Score 5: The response is completely correct, accurate, and factual.

###Feedback:"""
```

Fig. 7. Evaluation Prompt

Human evaluation We got 18 responses in total from students in SKKU, and he result of human evaluation is as follows: Our model got 4.38 out of 5.0 for overall performance, indicating the practicality of our model. Relatively low score for the latency is due to the wifi connection, not the latency of our model. Due to the recent heavy snow, the wire connection to the server got a breakdown, so we had to use the

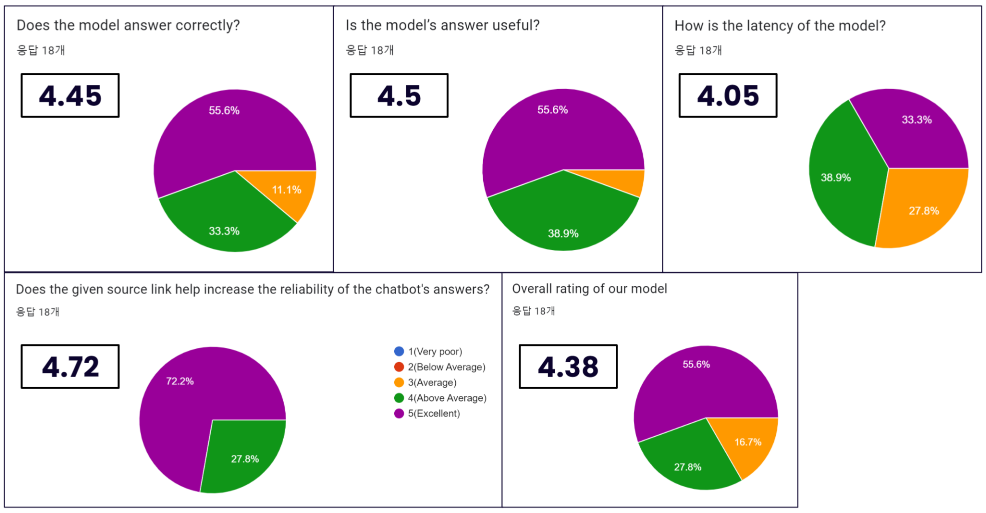


Fig. 8. Human evaluation result

slower wireless connection unfortunately. Providing links to the original contents was favored by most of the participants.

6 Limitation

Despite the promising performance of our RAG based chat-bat, it still has some limitations. First, the size of the database is small, not covering the whole information about SKKU. Second, the chat history is not saved per user, so it is refreshed every time a user leaves. Third, image information cannot be handled, since our chat-bot is only a text based model, not a multi model. Finally, our chat-bot always retrieves a document even if the query is not about asking about our school information-such as 'hello'-leading to failure in general chat situations.

7 Conclusion

It is challenging for the international students to get relevant information about our school, due to the language barrier and fragmented information. Existing works have limitations in monolingualism and key-word based search. For that reason, we developed a RAG based chat-bot that is capable of both Korean and English, and diverse queries. We tried to enhance the model's performance by adding the reranking process in the retriever. Also, prompt engineering and streamlining was implemented to enhance user experiment. We kept the database up to date by updating the database everyday at midnight. We hope that our work serves as a cornerstone for a stronger chat-bot model for SKKU.

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

A Appendix

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