CS/ECE 8803 Conversational AI Final Project Report - Reggie the Registration Chat Bot

Arjun Verma
College of Computing
Georgia Institute of Technology

Ethan Haarer
College of Computing
Georgia Institute of Technology

Jai Jain
College of Computing
Georgia Institute of Technology

Ege Gunal
School of Electrical and Computer Engineering
Georgia Institute of Technology

Abstract

This paper presents Reggie, a conversational AI tool designed to assist new Georgia Tech students with various aspects of onboarding and registration. Reggie aims to be an exceptionally knowledgeable chatbot capable of integrating both static and real-time data to answer personalized student inquiries. Through automated LLM and human centered testing, we find that Reggie offers an effective and innovative solution to improve student onboarding and provides timely, context-sensitive support for students adjusting to life at Georgia Tech. Our code can be found here `https://github.gatech.edu/averma332/CAI_Final`, and a live version of our chatbot can be accessed here `https://huggingface.co/spaces/RangDeBasanti/Reggie`.

1 Introduction

For incoming and returning students at Georgia Tech, navigating the registration system has always been a mountain to overcome. Much of the information about degree requirements, professor ratings, and available openings for classes are spread across official information pages, third party rating sites and the labyrinth of pages that compose OSCAR. On top of this, limited academic advisor availability leads many students without guidance or explanation. Approaching registration as a new student is intimidating, and missing any information could drastically impact what classes you end registration with, and even push back a student's graduation.

To aid students in overcoming issues with registration, we propose the creation of a Georgia Tech Registration Chat Bot named Reggie. Reggie will have access to a variety of registration information from the Georgia Tech registrar to proficiently answer frequently asked questions covering both general registration questions and department-specific information. Reggie will also have access to class availability and third-party site statistics about classes and professors to better inform students about their prospective choices. Reggie will also have the option to upload .pdf files of class syllabuses to extract core class information for easier comprehension of student expectations for classes as well as direct access to canvas for specific student class information. Finally, Reggie will have access to various different APIs related to class and registration information from Georgia Tech to load real time data.

All the information supplied to Reggie will then be accessible to its core LLM (Naveed et al., 2023) that has been supplemented through the use of RAG (Lewis et al., 2020) (Chivilkar et al., 2024) to load specific FAQs. This allows for a more dynamic interactions with users and how to handle the supplied information so students may use this service even when their advisor is unavailable. This allows students to easily compare and converse with this information to better comprehend the complexities of registration.

Throughout this paper we document the development and decisions made in creating the Reggie Registration Chat Bot. We delve into what documents work well with RAG versus which fall short, how we integrated API calls to synthesize information from across different websites, and how we account for variances in user inputs. In total, we create a unified chat bot system that conglomerates key registration and class information to students in a conversational approach (Dinan et al., 2018) to supplement the current difficulties brought about by registration at Georgia Tech.

2 Prior Related Work

We found multiple different papers that provided relevant insights into the process of developing Reggie both through personal research and the use of a case study. In finding modes to develop and train a chatbot with existing LLMs we look towards “Herding AI Cats: Lessons from Designing a Chatbot by Prompting GPT-3” (Zamfirescu-Pereira et al., 2023). This paper discusses the implications of creating a chatbot UX solely from prompting a modern LLM such as GPT3 or GPT4, namely that the process requires specific prompting and potentially lacks the ability to handle failure/edge cases from the user. As well as this, it discusses how to design a chatbot based on LLM prompting in a way that creates a satisfying and effective UX. The key ideas we hope to take away from this paper are the design principles behind creating a chatbot based off of a modern LLM such as GPT3/4 as that is going to be a core portion of our project.

In finding methods to supply information into our LLM we found the paper “Perplexed by Perplexity: Perplexity-based pruning with Small Reference Models" (Ankner et al., 2024) which explores methods to effectively pretrain smaller LLMs with available data by leveraging selective data sampling routines. Through the focus on domain specific data, such as registration information, this paper advises to minimize input information into frequent question-answer pairs to enhance model proficiency and maintain performance. The paper also addresses other methods to improve domain-specific fine-tuning, including parameter-efficient fine-tuning, and augmenting knowledge retrieval. These methods would improve our training for Reggie, especially given the Q & A dataset from advisors common FAQs. This would ensure relevant, high-quality responses that should also help to mitigate potential hallucinations regarding registration information.

In maintaining FERPA guidelines and maintaining student anonymity we found, "LLM Unlearning via Loss Adjustment with Only Forget Data", focuses on methods that adjust the external knowledge base of RAG without interacting with the model, effectively simulating forgetting (Wang et al., 2024). However upon further investigation, we found that ensuring that any data provided from students need not be used in RAG as this impacts model response clarity, so instead we chose to hold all user-fed data into chat history, this way individual student data is never stored permanently and only held in-session. This ensures no student data is held when conversations close, maintaining FERPA guidelines.

Reaching out to others that have undergone similar projects for their advice on how to approach this project, we spoke with PHD student William Gay who previously worked on an FAQ chat bot regarding the AE department at Georgia Tech. Speaking on his experiments in loading a model with GT-specific information, William stressed the importance that over-utilizing RAG and attempting to directly fine-tune the model only achieves middling results. He detailed that these fine tuned models often produced nonsensical or irrelevant responses to requests, and advised that it would be better to target integrating API functionality to retrieve real-time information from other sites rather than trying to overload the model on information pertaining to a singular topic like registration.

The need for real-time information in CAI systems was further stressed in the case study Knowledge Augmented Conversational AI presented earlier this semester (Chivilkar, 2024). In connecting these outside systems to a conversational system, one is able to integrate contextual understanding and real-time knowledge on domain-specific information, enhancing responses and overall utility of a chat system. Combining the context with user question allows a model, especially an LLM to interplay context from previous requests to craft more complex and individualized responses. Corroborating Gay’s findings, this case study emphasizes that solely RAG-based systems struggle to answer multi-hop questions and tasks, not being able to combine and synthesize information across multi-turn conversations. However, relying on external systems to supply information can fail if this information is useless or unclear to the model or user. Ensuring consistent formatting and context and accounting for errors or edge cases is paramount to building a robust knowledge-based CAI system.

This being said, RAG shouldn’t be completely discarded. From the Retrieval Augmented Generation case study from earlier this semester (Cho, 2024), RAG can help to prevent an LLM hallucinating facts, which will help with establishing baseline information about Georgia Tech as there is information that cannot be supplanted by integrating an API, such as core details about GT registration. In this way, integrating a limited selection of documents to be used in RAG should be the best case as we both provide context needed for the model while still relying on integrated exterior systems to provide real-time information to still allow for multi-turn dialogues.

There existing multiple techniques to evaluate

LLMs, the most prevalent ones to our case being automatic LLM assisted evaluation and direct human feedback. As described in Desmond et. all's work, LLM assisted evaluation consists of utilizing a question database of use-case questions and passing in the chatbot's response alongside each question to a secondary LLM to determine the quality of the response (Desmond et al., 2024). Direct human feedback can be done in a variety of different ways, including querying users through direct interaction or intelligently designed forms as described by Tam et. al (Tam, 2024).

3 Model

Our model pipeline consists of a base LLM (in this case GPT 3.5) that has been augmented with a number of different external knowledge bases to increase its functionality and response capabilities, all wrapped by a interactive UI. Specifically, our base LLM is augmented by a GT document RAG vectorstore, an integrated Canvas API, an integrated RMP (Rate My Professor) API, an integrated OSCAR API and an integrated specialization requirements API. The combination of the APIs and RAG allows for our model to accurately answer user questions that require both dynamic and static information regarding all facets of registration and onboarding at GT. When a user prompts our model pipeline, their prompt will follow the model workflow shown in Figure 1.

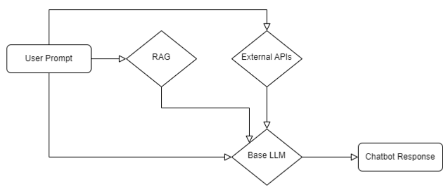

Figure 1: Chatbot Workflow Diagram

The following sections will cover each portion of the pipeline in depth.

3.1 UI and Base LLM

For the core LLM that will be the backbone of Reggie we chose to use OpenAI's GPT-3.5 model for two main reasons. First, we wanted to choose a model that robust and able to handle complex tasks, especially when provided wider context and information from the user to better tailor responses. Second, we needed an inexpensive model that could be affordable if consistently used by many students, especially during periods of high-traffic like phase 2 registration.

To increase the usability of this model beyond a command-line interaction, we utilized hugging face's gradio library to configure a basic chat web app. Using the gradio library, we could also keep track of the chat history separate from the chat history visible to the user. This functionality allowed us to insert texts into chat history like file contents and prompt engineering into requests without them being visible to the user.

3.2 RAG Integration

Our RAG (Retrieval Augmented Generation) is designed to expand the "memory" of our LLM to include relevant documents regarding multiple facets of registration at Georgia Tech. These documents include Registration FAQs, TA'ing FAQs, CS Core Requirements, and much more - we expound on this list further in our data section. Our RAG allows for our base LLM to access the information in these documents through a vector similarity search, where both the user's prompt and 300-word long portions of each document are encoded and matched by cosine similarity. More specifically, our RAG selects the three closest matches and augments the prompt with this information, allowing the model to collect relevant information from the docs based off the user's query. We found that 300-word portions and a search size of 3 returned the best results empirically. We implemented our RAG using LangChain, allowing for us to smoothly integrate it with our model pipeline.

3.3 Canvas API Integration

Our Canvas API integration is designed to allow Reggie to collect information regarding an end users classes and assignments - this will make Reggie more informed about the current workload of the end user and thus help it craft more personalized responses. The Canvas API provides multiple different endpoints to collect information from the website - the main ones we utilized were the "assignments" and "classes" endpoints. Querying these endpoints allows us to dynamically retrieve information regarding an end user's active classes and assignments.

We integrated this API into our chatbot using OpenAI's "Function Calling" feature, allowing Reggie's core LLM to decide when it would be useful to call the Canvas API. This allows us to seamlessly provide the model with data from Canvas whenever necessary.

Utilizing the Canvas API requires a Canvas access token - we were able to integrate this into our local model through the use of a private environment variable, however, we were not able to integrate it into our live version. Directly providing an access token to the model presents a huge security risk - we hope to handle this problem in the future by adding an OAuth2 layer to our model's Canvas integration. This would redirect users to SSO upon entering Reggie's website and thus allow Reggie to securely access Canvas.

3.4 RMP Integration

Seeking to provide users with third-party metrics about professors, we integrated the ability for Reggie to access metrics collected from the site RateMyProfessor.com and utilize these metrics in it's conversations. To integrate this API we utilized OpenAI's 'Function Calling' feature, allowing the LLM that makes the core of Reggie to decide when in a conversation it would be relevant to execute a specific function.

In this scenario, we describe our function to look out for the user seeking some sort of rating or judgment, alongside looking for a first and last name from a professor. This then passes these details to an integrated function where we call the RateMyProfessor API and search Georgia Tech professors for an exact match to the user query. Then a professor's rated quality and difficulty is then sent through the model for it to formulate an appropriate response to the user based on their overall query.

This functionality is not limited to single professors at a time, as we have tested it to work on multiple professors within the same user request to the chatbot. This further allows the model to compare the quality and ratings of the professors at the user's request, and help students make decisions on which professors they may prefer over others in a unified interactive system rather than having to skip around to different professor's webpages and comparing these statistics themselves.

3.5 OSCAR Integration

Our OSCAR integration is designed to allow the end user to receive real time information regarding the registration and waitlist status of GT courses. We achieve this by utilizing the GTScheduler API endpoint, which allows us to poll for the number of waitlist spots and regular spots open for a certain course given its CRN through a simple GET request. This information is very useful to potential end users because it would inform them regarding whether or not they could register for or waitlist for a certain course. We integrate this API into our model through OpenAI's function call API, which adds a portion to the LLM prompt that checks whether or not the user's request needs information from our OSCAR API and returns information relevant to our OSCAR API (namely the course CRN) from the user's prompt if necessary. This allows for us to smoothly integrate the OSCAR API into our chatbot, allowing for users to weave complex queries and receive responses with real time information.

3.6 Specialization Requirements Integration

One facet we wanted to integrate into our chatbot is the ability for Reggie to retrieve what class information different students need to take in order to graduate. However we were faced with two main issues. First, unless we have permission from administration, we cannot access individual student's registration and major requirements without facing potential legal issues. Second, supplying class information into RAG often leads to confusing errors from the LLM due to it's probabilistic nature. For example, when supplied a document holding information on all classes, we asked Reggie what class for first year CS students primarily focused on the basics of python, its claimed CS 1332 is the correct answer, when in reality, CS 1301 is the correct class. After repeating these tests for multiple classes we found that the similar naming schema of classes led to model confusion when mapping exact classes to their descriptions, leading RAG to be insufficient for this task requiring pursuing a different avenue of implementation.

To resolve this issue, we sought to utilize OpenAI's function calling feature, similar to how we implemented the Rate My Professor functionality earlier. First we began by conglomerating all classes needed for a specific degree. We chose to focus on Georgia Tech's MSCS program as half our team is currently in this program and it is a solid baseline to build from for other majors. Using program information we created a function that took in a major and a specialization. The OpenAI API would scan user responses for cases containing these two parameters, and judge the prompt to see if the user was indeed asking for required class information. From there, the function processes the imputed parameters, taking into account common abbreviations for majors and specializations to account for in-formalities and ease of use for user responses. Based on the major and specialization, the model returns the different required courses and lists of courses to choose from to complete any specialization.

Then, it was a simple implementation to integrate this function into the existing chat bot. Although this only covers a single degree program, with additional time it would be possible to integrate other degree programs and majors to serve a wider variety of students. Doing so could potentially allow students to compare different majors and could further help with questions students have while changing majors, including which classes may count for multiple programs.

3.7 Multi-modal Syllabus Ingestion

When switching between classes, it is often difficult for new students to parse through syllabi themselves and find what sections hold key information like grade distribution, Assignment weighting, late policies, and key course content. To help students efficiently deal with multiple syllabi at once, we wanted to integrate some for to summarize these key points, while still providing students the opportunity to converse with Reggie about course specifics. To do this, we sought to integrate multimodal inputs to our chat bot to upload .pdf files to be processed by our chat bot.

We first started by modifying our current input to our LLM by exchanging our input bar to a multimodal one supplied through the gradio library. Any uploaded file is separated from regular text in the chat history by being assigned a special 'file' role which is then used in processing. Then for every file uploaded we processes each separately, allowing for multiple inputs at once. Using the mimetypes library we identify the file type, and allow it's content to be sent through to the LLM, otherwise we discard the file. We then scrape the text data from the .pdf file, and hold the original content in the chat history such that it is still accessible to the LLM but not visible to the user.

We then pass through the user query and the content of the file through to a specialized prompt pipeline separate from the Reggie persona. This pipeline first determines whether or not the syllabus is actually a syllabus. If it is, then the LLM extracts the key information and return the found information to the user. If the LLM determines the file contents are not a syllabus, a specific rejection message is sent to the user and the contents are removed from the chat history so it is not accessible to the chat bot for future queries.

Through this implementation, we both increase the ease-of-use for interacting with our chatbot by reducing the steps needed to supply the bot with outside information, as well as pave the way for future iterations of this chat bot to utilize multimodal inputs in different ways, expanding the utility of this chatbot. Similar to integrating the ratemyprofessor API, adding this information to the chat history allows the chatbot to dynamically interact with the content within these files, and even compare different classes syllabi to help advise students on which class is best for them.

4 Data

4.1 RAG Data

As stated previously, our RAG consists of multiple documents that contain information relevant to our model. These documents include but are not limited to Registration FAQs, TA'ing FAQs and CS Core Requirements. A full list of the documents selected can be found here: https://drive.google.com/drive/folders/1iZCSHf5s2k7eAZUv0NvuJtqU3EdIGqi_?usp=sharing.

In selecting what documents would be practical to establish a baseline for our model, we were first directed to several FAQ documents William Gay had used in his testing, which held question-answer formatted text directly from Georgia Tech advisors which greatly improved the knowledge base of the model. We then sought out similar documents that had demarcated information into smaller section, such as brief sentences, bullet points or Q & A format. This was mainly done as during testing, documents with information that was separated out instead of held in long for paragraphs or dispersed throughout a single document led to more consistent and correct recall of facts during generation.

In total, we landed on seven moderately sized documents, most of which were either provided documents from GT or from GT webpages. We found that any more documents led to further confusion for the model and negatively impacted multi-turn recall of information, with the model often forgetting information it had just previously sent in an earlier response. This way we balance supplying the LLM with a baseline of primary information with ensuring the bot maintains the autonomy to aggregate information from prior messages and third-party sources.

5 Experiments

Our experiments consist of two main portions - automated LLM evaluation and human evaluation. For both methods, we utilize OpenAI's ChatGPT as our baseline - our goal is to outperform it.

5.1 Automated LLM Evaluation

Automated LLM evaluation consists of utilizing a secondary LLM to grade the response of a base LLM based off a given user query, using the LLM as a Judge method (Huang et al., 2024). Our automated LLM evaluation experiment is designed to cover a breadth of queries that span multiple different potential subjects a end user may bring up with the chatbot. These subjects include but are not limited to questions about where to register, specific questions about course waitlists, specific questions about professors, and questions about how to handle registration errors, as well as questions extracted via another LLM based on a given set of documents. A list of our selected fixed questions can be found here: `https://github.gatech.edu/averma332/CAI_Final/blob/main/LLMEval.py`.
In order to evaluate our chatbot against a baseline on these questions, we developed a prompt that allowed for our secondary LLM to accurately grade the responses of both our chatbot and a base LLM (GPT-3.5) on a scale of 1 to 10 based on metrics such as accuracy, relevance and completeness.

5.2 Human Evaluations

Our human evaluation experiments are designed to extract complicated conversation flows from our chatbot and grade them based on a human evaluation metric. For our human evaluation experiment, we randomly sampled 20 students from Georgia Tech and asked them to ask a series of relevant questions to both our chatbot and ChatGPT and grade the responses of both systems on accuracy in response, helpfulness of response, and ease of use. We distributed these questions as a survey through google forms spread over slack channels and random emails.

6 Results and Analysis

6.1 Automated LLM Evaluation

Overall, we found that our chatbot outperformed the baseline ChatGPT model in terms of average score given by our evaluator LLM. The scores can be seen below:

	GPT-3.5	**Reggie**
Average Score	5.8	7.25

Table 1: Comparison of GPT-3.5 and Reggie LLM evaluations

Reggie outperformed the base GPT-3.5 when judged on metrics like accuracy and completeness by an external LLM, making it a better option for end users based upon those metrics. However, simply looking at the average results does not provide a fine-grained insight into the benefits the chatbot has compared to the base LLM. We will now consider a few summarized results from the evaluation process that provide interesting insights. First, we will consider the following:

Clearly, Reggie provides a more relevant answer because it is able to accurately query OSCAR through its OSCAR API integration and provide a real number of seats available instead of acting like the base LLM and telling the user to do so. We can consider another example in Table 3:

Both the base LLM and Reggie perform poorly on this question - the base LLM provides a heartfelt and detailed answer with no relevant data and Reggie provides a colder and less detailed answer with relevant data. This highlights how the base LLM can be easier to use and talk to, but less relevant and accurate.

6.2 Human Evaluation

Overall, we found that our chatbot outperformed the baseline ChatGPT model in two of the three categories, the exact results of which can be seen in Table 4.

To receive this data, we had 20 participants ask both chat GPT-3.5 and our Reggie chat bot questions about Georgia tech, registration, and other specifics of their choosing to each bot. Each participant completed a form where they ranked the

Are there available seats for the CS 3600, CRN 30312?	
Base ChatGPT	
Response	**Grade**
As an AI, I do not have real-time information on course availability. I recommend checking the university's course registration system to see if there are available seats for CS 3600 with CRN 30312.	2/10: The chatbot's response is partially correct but lacks helpful details, as it doesn't provide guidance on where to find course availability information. It could have directed the user to the official course registration system or offered instructions on checking seat availability, resulting in a low grade.
Reggie	
Response	**Grade**
Based on the information provided, there are currently 3 seats available for CS 3600 with CRN 30312.	9/10: I would rate this response a 9 because it directly answers the question by providing specific data on the availability of seats for CS 3600 with CRN 30312. The response is clear, concise, and addresses the query effectively.

Table 2: Question and Responses with Scores

Should I be a mscs student? Idk if I can handle too many courses?	
Base ChatGPT	
Response	**Grade**
Deciding whether to pursue a Master's in Computer Science depends on your strengths, time management, and dedication, and it's important to prioritize your well-being and academic success when choosing your course load.	3/10: The response is vague and doesn't provide specific data or guidance. It lacks detailed information about the Master's in Computer Science program at Georgia Tech, such as the typical course load, resources available for support, or success rates of students with different academic backgrounds.
Reggie	
Response	**Grade**
As an MSCS student at Georgia Tech, you will need to be open and flexible to taking courses that fulfill your specialization requirements. It is important to keep in mind that there will be a Fall 2024 Phase II registration in August. If a course you want is full, advisors cannot add or remove students, so it is important to be patient and willing to explore other available options for your degree requirements.	3/10: The response provided by the chatbot did not directly address the user's concerns about potentially being overwhelmed by too many courses as an MSCS student. Instead, it focused on Phase II registration and advisors not being able to add or remove students from full courses. The response lacked helpful advice or guidance for the user's specific situation, resulting in a low score.

Table 3: Question and Responses with Scores

helpfulness, accuracy, and ease-of-use on a scale of 1 (poor) to 10 (excellent). Each participant was asked to send at least five different requests (not including clarifying messages) to each bot to get a feel for the functionality, knowledge and utility of each system.

When asking users after their survey why they ranked one chatbot over the other, users found the relevancy and depth of information provided by Reggie to outperform that of regular chatGPT. In general, Reggie was able to more consistently provide relevent links and key information than chatGPT, which often either hallucinated information, or simply told users to refer to related administration sites, often without providing the relevant links that Reggie would. However, they also cited that chatGPT often required less clarification in certain requests, leading to easier usage overall.

To illustrate the effectiveness of our system, we randomly sampled one of the conversation flows from our survey. We found that our chatbot tended to provide much more relevant and accurate answers to more complicated questions whereas Chat-

	GPT-3.5	Reggie
Helpfulness	5.1	7.6
Accuracy	4.7	7.45
Ease-Of-Use	8.15	6.3

Table 4: Comparison of GPT-3.5 and Reggie human evaluations

GPT did not have the necessary information. An example of which can be seen in figure 2, where the end user asks for live information regarding a class:

Clearly, our chatbot is able to provide a more relevant answer than ChatGPT. However, ChatGPT's flow and ease in conversation is smoother. Another example can be seen in figure 3 where an end user asks for information regarding a class:

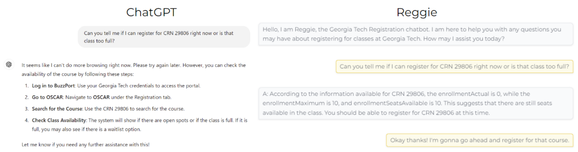

Figure 2: ChatGPT vs Reggie - Registration Question

Clearly, our chatbot is able to provide a more accurate and informed answer than ChatGPT. However, our chatbot tends to be more curt whereas ChatGPT responds in a "nicer" manner.

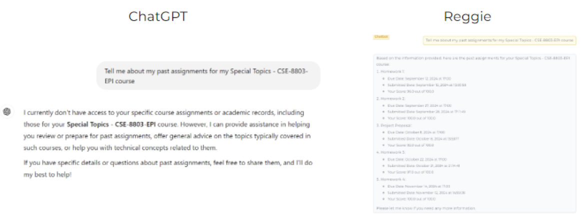

Figure 3: ChatGPT vs Reggie - Class Question

7 Conclusion and Future Work

In developing Reggie, we sought to not only aggregate the resources needed to navigate the Georgia Tech registration environment, but do so by integrating this information into a conversational format. By integrating static knowledge through RAG and dynamic API functionality, Reggie provides accurate, contextually relevant, and actionable information to users, addressing a critical gap in student support systems.

Our evaluations show that Reggie consistently outperforms the base GPT-3.5 model in accuracy and helpfulness, and with further work Reggie can improve its conversational flow to achieve the sophistication of the GPT model intractability. Reggie represents the potential of combining LLM technology with thoughtful application design to overcome common challenges in modern academia. By enabling a system to dynamically interact with students requests for the information, we enable students to make informed decisions about classes, professors, and class availability throughout the registration process.

In the future, we hope to improve Reggie in a multitude of ways. We plan to increase the multimodal aspect of the chatbot, integrating picture and video input to allow Reggie to process and provide insight more accessibly. As well as this, we plan on making the Canvas API aspect truly accessible, adding an OAuth layer to the chatbot to allow for all students to access each of their individual canvas datum. We also plan on reaching out to GT to access the official resources required to deploy such as chatbot (school approval, access to more data, etc.). Additionally, we hope to improve Reggie through optimizing its RAG. Building a question and answer dataset and fine tuning the RAG embeddings such that questions are matched perfectly to their answers through cosine similarity loss or another similar metric would make Reggie's responses regarding FAQ's much more accurate. Future iterations of Reggie could expand its capabilities by incorporating additional degree programs, improving conversational finesse, and integrating advanced personalization features to make interactions even more seamless and engaging.

By simplifying complex processes like course registration and providing timely support, Reggie not only empowers students to navigate their academic journey more effectively but also sets a benchmark for future advancements in conversational AI for higher education.

References

Zachary Ankner, Cody Blakeney, Kartik Sreenivasan, Max Marion, Matthew L. Leavitt, and Mansheej Paul. 2024. Perplexed by perplexity: Perplexity-based pruning with small reference models. In *ICLR 2024 Workshop on Understanding of Foundation Models (ME-FoMo)*.

Shreya Chivilkar, Twisha Shah, Taehyuk Park, and Kasmira Kurnala. 2024. Retrieval-augmented generation for conversational ai. Case study report, Georgia Institute of Technology. Available upon request.

T. Park T. Kurnala K. Chivilkar, S. Shah. 2024. Case study - retrieval augmented generation. [PowerPoint presentation].

H. Yi S. Zhou X. Cho, M. Kim. 2024. Case study - knowledge augmented conversational ai. [PowerPoint presentation].

Michael Desmond, Zahra Ashtorab, Qian Pan, Casey Dugan, and James M. Johnson. 2024. Evalullm: Llm assisted evaluation of generative outputs.

E. Dinan et al. 2018. Wizard of wikipedia: Knowledge-powered conversational agents. *arXiv preprint*.

Hui Huang, Yingqi Qu, Jing Liu, Muyun Yang, and Tiejun Zhao. 2024. An empirical study of llm-as-a-judge for llm evaluation: Fine-tuned judge models are task-specific classifiers. *arXiv preprint arXiv:2403.02839*.

P. Lewis et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2023. A comprehensive overview of large language models. *arXiv preprint arXiv:2307.06435*.

Sivarajkumar S. Kapoor S. Stolyar A. V. Polanska K. McCarthy K. R. Osterhoudt H. Wu X. Visweswaran S. Fu S. Mathur P. Cacciamani G. E. Sun C. Peng Y. Wang Y. Tam, T. Y. C. 2024. A framework for human evaluation of large language models in healthcare derived from literature review.

Yaxuan Wang, Jiaheng Wei, Chris Yuhao Liu, Jinlong Pang, Quan Liu, Ankit Parag Shah, Yujia Bao, Yang Liu, and Wei Wei. 2024. Llm unlearning via loss adjustment with only forget data. *arXiv preprint arXiv:2410.11143*.

J.D. Zamfirescu-Pereira, Heather Wei, Amy Xiao, Kitty Gu, Grace Jung, Matthew G. Lee, Bjoern Hartmann, and Qian Yang. 2023. Herding ai cats: Lessons from designing a chatbot by prompting gpt-3. In *Designing Interactive Systems Conference (DIS '23)*, pages 2206–2217.