LIGN 167 Final Project Writeup

Introduction

For our project, we created an AI Tutor for the class LIGN 167 that will help tackle the majority of problems that students might face when learning in this class in the future, even with Chat GPT here to help them. We agreed that the lack of resources specifically tailored to student needs was our major problem to tackle. Every student learns differently, whether through physical printed flashcards, practice quizzes, or cheat sheets. Whether it's creating printable flashcards, practice quizzes, or even cheat sheets that students would like to use to learn, Chat GPT-4 can be limited in its content creation capabilities. Furthermore, as it does not have access to class materials, it does not provide advice that does not necessarily align with course learning objectives, and material tested on assignments, projects, or exams. For example, if students have questions about a particular homework problem and want to know what relevant lecture to watch, or they want the mathematical notation provided by the bot to be consistent with what is used in the Textbook and Lectures, Chat GPT-4 is insufficient. We leveraged Generative AI to produce a helpful chatbot, flashcards, cheatsheets, and practice exams that are tailored to students' needs and feedback. Each of us was responsible for working on a particular feature of the project: Lakshmi worked on the chatbot and the flashcards, Michael worked on the responsive practice exams, and Kevin was responsible for the cheat sheet generator.

Results

Chat Bot:

We used the class resources, namely the textbook, syllabus, and problem sets to improve the answers generated by Chat GPT, and create a chatbot that students could use. To overcome the input limits we created a knowledge database using text embeddings from these resources, and used similarity search to generate relevant text. Then we created a simple UI to allow the user to ask questions to the bot. When the user asks class-specific questions like "What lecture should I watch to solve problem number 5 on Problem Set 2?", or "When is Problem Set 3 due?" The bot produces satisfactory answers, something that Chat GPT-4 cannot do on its own. Some things that I would improve about the bot are that it's very direct, especially in comparison to using a regular version of Chat GPT-4, which is more verbose and "kinder". The largest issue when making the chatbot, since this was the last feature we implemented, was to incorporate the Python code into the web app using Flask. Web development was a completely new field for me, so there was a much larger learning curve in comparison to simply using Javascript and HTML. We made the executive decision to present this as a standalone feature rather than modify our entire code thus far. However, with the use of Python libraries like langchain, and faiss, it was relatively easy to create the text embeddings from the class materials.

Practice Exams:

We found that in the process of generating practice exams, specifying whether a question should be conceptual or computational was necessary as GPT-4 tended to generate vague conceptual questions if this prompt was not included. In addition, the background provided in the system content section of the messages sent to GPT-4 were crucial in ensuring that the questions were answered as accurately as possible (i.e. specifying the field that the question covers makes the GPT-4's feedback on given answers much more accurate). Much of the testing was conducted with GPT-3.5 as the GPT-4 API response times were far too slow for practical use, despite GPT-4 generally yielding better results.

One of the most difficult aspects of attempting to incorporate GPT-4's capabilities into an existing application was prompting it to format its output in a specific way. Generally, purely semantic demands (i.e. include 5 questions on backpropagation) were pretty easy to follow; however, structural prompts (i.e. divide questions using custom separators, mix up questions so that they are not grouped by topic) yielded much less consistent results. This may be a result of the OpenAI models being trained to focus more on quality of output rather than its structure. Additionally, we found that parsing LaTeX from both GPT-4's outputs and within the website was extremely difficult and we were unable to resolve this issue. Given a longer time frame, this would be one of the first things we improve since it is important that the information is presented in a way that is easily readable.

Flashcards:

We used a similar approach to engineer a good prompt to generate flashcards. We found that it was key to emphasize for GPT-4 to stay on topic with the material within the topics chosen. It was also important to our code for GPT to format its response a particular way, but my code handled different types of inputs and formatted the text to create the flashcards. One thing I would have added, if we had time, is for the student to be able to download a printable version of the flashcards, or an anki file to be able to upload the flash cards to Anki for spaced repetition purposes. These ideas were also suggested to us by other students when we were presenting our project. The response time for GPT-4 was quite slow, taking far more time than desirable, however when we used GPT 3.5 the responses were of similar quality, but with much more reasonable times. In the future, I think GPT 3.5 is sufficient for this feature.

Cheat Sheet:

The AI tutor project for generating cheat sheets has shown notable success in creating a user-centric learning aid. Its ability to process and produce intricate mathematical formulas and concepts tailored to the topic of cheat sheet generation marks a significant achievement. Users benefit from the versatility of the system, having the freedom to select from a diverse range of topics provided by the syllabus and still receive a customized cheat sheet. The interactive element, where users can provide feedback, ensures a user-oriented approach to learning, allowing for continuous improvement of the cheat sheet content. Moreover, the project facilitates

educational accessibility by enabling users to download their personalized cheat sheets as PDFs, a crucial feature for offline study and print.

Conversely, the project faced several challenges that provide opportunities for further refinement. One of the primary issues was the necessity to extract and convert LaTeX formatted elements individually to ensure the text's readability, which is a labor-intensive process that could be streamlined with more sophisticated parsing algorithms. Additionally, the difficulty in creating line breaks or appropriate spacing in responses was notable; this was partially mitigated by employing prompt engineering alongside HTML. However, the constraints of PDF formatting proved more arduous than anticipated, suggesting a need for enhanced PDF generation techniques. The interaction with the AI model also highlighted the limitations in terms of the word count per interaction, which necessitates careful content planning. Lastly, the cost and complexity associated with fine-tuning the AI's performance for optimal results present ongoing challenges that require strategic resource allocation and innovative problem-solving approaches. However, we decided not to implement the course materials directly into the cheat sheet because this class is a class that shows the current trend and history in deep learning and a large language model. It would be more accurate and efficient to use prompt engineering for creating cheat sheets to implement the latest news or algorithms to meet the needs. Since we have not had midterm or final exam for LIGN 167, there was no ground truth to compare and test performance accuracy of our model. For real uses, we can change to whatever is the latest large language model with new topics of interest that aligns with future syllabus.