William Kam

Gina Roberg

Anthony Yao

Project Writeup: Note Sheet Generator

In our project, we created a web application that allows users to upload class content in the form of pdfs, to generate a customized note sheet. Once users upload PDF documents, a GPT call is made, using the OpenAI API, to extract the text, analyze the text, and derive core concepts from the text. The user is then prompted to select what topics they would like to be included in the note sheet. Once the topics are selected, the note sheet can be generated and subsequently downloaded as a PDF by the user. Afterwards, if the user finds any aspect of the content dissatisfactory, the user can then input feedback to make another OpenAI API call and the pdf will be regenerated based on the feedback for the user to download.

The use of OpenAI's APi for text analysis and concept extraction worked effectively, providing a structured summary of PDF content. We used prompting to ensure that the structure of each summary of each lecture would have the same format and contain all the content that we wanted. This was necessary because the way we built the pdf files required common structure in our outputs so we could use regex to identify core concept headings as well as creating our pdf. In addition, our app allows for customizable PDFs based on user selected topics and our app also has the functionality of dynamic font and space sizing.

Some things that didn't work out too well was our processing speed. For each lecture, we had to call OpenAI Api once which wasn't time intensive if we only had a few pdfs but for uploads with larger and more pdfs or processing time went up significantly. In addition, we had trouble with how we were storing our outputs from OpenAI API. At first we were storing all of our outputs as lists but then we realized that we couldn't map our content to the names of the pdfs. So then we had to modify all of our functions to work with a dictionary instead. In addition, we also had some trouble displaying content when a user selects a few topics. We found that sometimes when users select topics, some of the topics don't show up on the cheatsheet. The problem was that our regex wasn't matching correctly, to address this we added a new line character after each core concept which fixed this bug.

One of the major things we had to learn was React. Our group was fairly inexperienced with React so we had to essentially learn on the go. Creating components was the easy part, but connecting our python backend to our frontend pieces was quite challenging. First, we had our python backend end code stored in a jupyter notebook so we had to convert our code into a .py file using flask for api development. We had to do this so we could handle file uploads and serve dynamic content. Furthermore, we also had to figure out how to configure CORS in Flask which is essential for enabling the frontend to communicate with the backend securely. In addition, we also had to take some time to figure out how to route to a second page after a user uploaded all their documents.

Another thing that we learned was both how powerful NLP integration can be but also their limitations. NLP integration can be very powerful in language processing and analysis but this integration can only handle so much data. We initially tried to combine all uploaded content into one text and send it to OpenAI API but there were too many tokens. This limitation resulted in us having to process each pdf individually, clean up the output, filter the output, and then merge into a pdf while being able to process all the information at once could have saved us some time on these steps. In addition, we also had no experience with pdf manipulation particularly libraries like pdfplumber and report lab so had to learn that while doing this project too. Another thing that didn't work too well was security vulnerabilities with our API key. In our code we hardcoded our API key which isn't secure as anybody who can view our code has access to this sensitive information. Moreover, we also had trouble with error handling limitations. For example, if a user uploaded a file that wasn't a pdf the most we could was send an error or when API requests encountered issues we couldn't really do too much. Some things that did work well in our project was our website. Using react, we were able to create a clean and user friendly user interface that had a functional frontend and backend. The last feature allowing the user to edit the outputted sheet with chatgpt is also slightly flawed, sometimes, chatGPT will not format the it's response properly and our pdf parser will not be able to read the edit. As such some edits were not displayed. A main improvement to our project that we could have implemented is asynchronous processing which could significantly improve performance when handling multiple files. Our group is proud of the fact that we were able to pick up React, Flask, PDF processing libraries, and OpenAI API to create a web application that is able to handle a vast variety of course content and be able to generate a useful cheat sheet based on the uploaded content.