Financial Fun Chatbot

Introduction

For our course project this semester, we were asked to create a chatbot that could extract financial information about two companies' 10Ks, and then allow the user to search and ask questions about the information within the file. The aim was to create a program where the user could use natural language to ask about the parts and items within the 10Ks, while also saving session statistics and logs, allowing the user to ask questions about these as well. I chose to approach this using Python, as I wanted to challenge/force myself to learn one of the fastest growing languages, especially during this age of AI and Machine Learning. Python also has a vast amount of well supported languages, that I knew would make it easy for me to scrape and parse files and deal with natural language. Scrapers/projects like this are extremely common, especially automated LLMs that allow financial analysts to analyze financial forms for investment purposes.

Solution

This project's required scope was to fully support the parts and items from the 10Ks of two chosen companies. I chose to do this with Alphabet Inc(Google) and General Motors, as they are two of the biggest companies in their respective sectors. The code is split into 4 main modules: the CLI/UI, FileReader, FileParser, and SessionLogger, with subsections within the FileParser to search the file and the CLI to map user intent with the supported queries. The user only interacts with the UI, with the other modules imported as packages into the CLI, where they are called after the user's intent is matched.

The first component created was PA1, which was the FileReader/Extractor. This module/component's goal was to get each company's 10K file and allow the user to view and ask statistics about it. I achieved this by using the BeautifulSoup Python library, which allowed me to parse a given website's html. Using this, I was able to extract the text from each of the company's SEC websites and then save it to a .txt file with the date and company. The original intent was to allow the user to search the SEC's website for any 10K, however that was both out of the scope of the project and would have required me to rewrite my code to use Selenium, a completely different library.

The next component created was PA2, the file processor. The goal of this component/module was to parse the file for its items and parts. I did this with the use of regex's, which allowed me to search for the start and end of each file's table of contents. This allowed me to parse through the TOC, dynamically creating a list of items and part regexes. These regexes were then saved to a list that the search function uses to find and return the part/item in the file. The search function sets the starting regex as the given part, with the end regex set as the next item in the TOC. The function then searches for the second instance of the starting regex(first instance is in the TOC) and returns all the text from that point until the ending regex is reached. This component also had a limited user interface, which I implemented through a CLI with 3 commands. These commands allowed the user to search through a part or item in a file or view the TOC for a given company.

The third component created was PA3, the chatbot UI. This code aimed to replace the limited and rigid CLI of PA2 with one that could take in user questions and then parse the file. I once again used regexes to parse through the user's input, checking to see what part or item of the 10K was in their utterance, and then calling the respective search method to retrieve it. This code also supported viewing the TOC of each company and the entire file in each case. This code was relatively effective; however, it did not allow the user much flexibility with partial matches and misspellings.

The fourth, and biggest, component created was PA4, the user intent to query mapper. This package was the most important and difficult, as it was supposed to allow the user to enter any natural language query and match it with one of the supported ones. This was difficult because my PA2 dynamically created queries, and thus I had to create them and then convert them to regexes. To achieve this goal, both the queries and user utterance are both stripped of special characters and lowered. They are both then split by word, allowing me to check how many common words they had. A built in Python Levenstein distance algorithm was used, which allowed slightly misspelled words to still count as common words. Once the percentage match was calculated, the code prompts the user to confirm the result, and then the appropriate part/item is retrieved. During this implementation, I also slightly modified the code of PA2 to call once and then store the individual parts and items in a dictionary. This improved performance, as the file was only read once rather than each time the user asked for a part/item.

The final component implemented was PA5, a session logger that aimed to allow the user to view session logs and statistics. This code was extremely easy to implement, as it required adding a couple lines to PA4 that tracked the time of a session, and number of user and system utterances. The code was also modified to write each input/response to a session log, and the session statistics to a csv at the end of each session using the built in Python CSV logger. Then, I created a CLI with two commands that allowed the user to view a summary of an individual session's statistics, all sessions, or view a session in its entirety. This was easy to implement as well, as I already had experience implementing a CLI for PA2 and parsing files for PA1.

Integrating all the components was a slight challenge, as I had used different UI methods for each part, especially for PA5. PA's 2,3, and 4 were all continuations of one another, so they were already integrated, and PA1 was simply a method that I imported as a module, which is called during the construction of the FileParsers. However, PA5 was difficult to implement due to the CLI commands I had implemented. Because of this, I would have had to redesign its' UI to work with my PA4. This is why I decided to reuse Dev Patel's PA5 UI and session log retrieval methods from his PA5, as he used regex questions in his UI rather than CLI commands. This made is much easier to combine my PA4 UI with PA5, as I simply had to add the sessionlogger queries to my PA4 list of supported queries.

The best component of my PA6 is PA2, as it dynamically creates and searches for the parts and items of each 10K. Many of the other projects in the class hardcoded the parts and item regexes, which makes it very difficult to update and reuse with other companies. However, my code can be used to parse through any 10K, as it can handle any combination of parts and items. For example, Google's 10K does not contain Notes, while GM's 10K does. My code can handle the additional parts of GM's 10k, as it is able to find and return these notes. This is also futureproof, as any updates, additions, or omissions to the companies' respective 10ks will be automatically reflected in the code.

Evaluation

Overall, I am happy with my PA6 and feel I met all the requirements of the project. My PA1 is the only one in the class that can web scrape each company's 10k, rather than manually adding the files. My PA2 dynamically gets the parts/items, increasing its reusability and future usability. My PA4 is able to handle almost any user request and match it to a part/item and my PA5 allows the user to view statistics and session logs, with the help of Dev's PA5. However, my

PA4 is slightly limited, as it handles chitchat by always matching the user's utterance with a default query, while other chatbots can handle simple chitchats unrelated to the 10k files. It also does not allow for the retrieval of multiple actions within the same query. For example, the user cannot ask for both Part 1 and Part 2 in the same utterance, as only Part 1 (or the first one) will be retrieved. The code also only runs in the terminal rather than with a dedicated UI, and thus looks ugly and cannot handle long responses as well.

Discussion

This project was extremely helpful at teaching me the basics of creating an AI chatbot, especially with the intent mapping of PA4. I also learned a lot about good coding practices to help with reuse, such as clear and consistent comments and detailed documentation. Testing was a big component of this project and a skill I lacked before, and thus it was extremely helpful to be forced to rigorously test my code and document it. I also learned Python throughout the project process, which has helped me create programs for other research projects and learn how to webscrape. I had an extremely positive experience building the solution, as it was extremely fun and rewarding to solve the problems and create solutions to the challenges posed by the 10k files. Learning Python was also extremely rewarding, as it is one of the most popular and used languages in the modern world of Computer Science. The majority of my collaboration occurred during the reuse process of PA6, as I had to communicate with Dev on how best to implement and reuse his PA5 code and how best for him to implement my PA1 code. This helped me realize the importance of good code documentation and comments.

If I was to continue working on this project in the future, I would first implement the planned search feature for my PA1 webscraper using Selenium. The goal of this would be to allow the user to input any company name or stock ticker, which the code would then search for in the EDGAR SEC database. The code would then get and parse the URL of the 10k file using the existing PA1 code. This would allow the user to search and parse through any 10k file, as my PA2 is already compatible with any 10k file. I would also try to create a GUI for my PA4, which would allow the user to interact with the program in a much nicer, cleaner way, while also preventing long responses from being cut off in the terminal. In this process, I would also try and clean up my PA4 code to make it more precise and adaptable.

Conclusion

Overall, I have learned a lot about good coding practices, Python, reusability, and chatbots during my time coding my Financial Fun chatbot. My PA6 was able to fulfill most of the project's requirements, and my code was reused by a multitude of people in the class. My code improved throughout the process, especially my code documentation and structure, and it's extremely easy to reuse due to its dynamic capabilities. With some modifications to my PA4 and 1, my code can be used with any 10k and company. This was one of the biggest and most difficult projects I have done individually, and I am proud of the result of my code. While there are some things I can improve and would improve in the future, I am happy with what I achieved in the end.