Codie Lee Munos

Research Assistant Agent

ITAI 2376

The agent of discussion for my report is based on the job role of a research assistant. At the beginning of the project, I felt very overwhelmed when reading the description of what was required. Some of it felt like I needed to redo the entire course again, but then I started to take it step-by-step in order to fully grasp what was being asked to accomplish. At first, I attempted to work with Azure studio to create my agent. This route did not work in my favor, therefore I went to where I was familiar. That ended up being Google Collab. Using this platform went a lot smoother personally and that is when I ventured into more depth of the goal of the assignment. I figured if I knew where to start and end, then I could start filling it in with more information.

With the overall experience I have had, I knew I needed to find out which libraries I needed to install to accomplish this project, at the least the starting libraries. I did have to install a few more throughout the project, but that was to be expected.

Looking at the Agent Architecture, I started with input processing requirements. I decided on my two sources that I would be using to determine what needed to be installed and downloaded. I picked Gemini AI Assistant and Kaggle for my dataset. Both of these required API keys. I had trouble getting the API keys to work at the beginning, and it honestly took a few hours to troubleshoot my issue. It became frustrating, but I took a small mental break and focused more on what dataset I wanted to use from Kaggle. I have worked with kaggle a few times, and this time found out, at least for me, that I need to keep only one copy of the kaggle.json file at a time due to expiring API keys I am assuming. After deleting my most recent install of the key, it suddenly worked. Then, it was time to upload my chosen dataset which gave an overview of inequality in education data.

After installing the Kaggle API Key, I moved on to the Gemini API key. This one required me to get a Google AI Studio account to access the key. Once I signed up, I was able to

locate the key and upload it to my Collab. I used the “key” section on the platform to save my keys to initialize once that code cell is run. This was something new to me as I have always kept the API key to be uploaded per run. This will help me save time for future projects!

Once the API key is initialized, the rest of the code cells requiring the data from either Kaggle or Gemini can run. This agent is able to use Gemini AI to answer questions about the dataset from Kaggle. Integrating the two different sources was a bit tricky, but I managed to figure it out. I did this by running the following code:

```
def answer_question_with_gemini_citations(query: str, dataframe: pd.DataFrame):
dataframe_string = dataframe.to_markdown(index=False)
    prompt = f"""
    You are an AI assistant that answers questions based on the following data.
    The data contains information about different schools, including their type,
    funding, test scores, and demographics.
    Data: {dataframe_string}
    Question: {query}
   Please see sources for more information.   """
```

Moving forward after the integration helped me gain more confidence in what I was doing. It was game time for the memory system to be deployed with the following code:

```
from typing import List, Dict, Tuple
def answer_question_with_memory(query: str, dataframe: pd.DataFrame, history:
List[Dict[str, str]], max_history_tokens: int = 1000) -> Tuple[str, List[Dict[str, str]]]:
    """
```

Answers a question using the Gemini model with conversational history and data context.

Args:

query: The user's question.

datafra

me: The pandas DataFrame to provide as context.

history: The list of previous conversation turns.

max_history_tokens: The maximum number of tokens for the conversational history.

Returns:

A tuple containing the model's response text (with citations if available)

and the updated conversation history.

"""

This cell was followed by def build_promt_with_history(history: List[Dict[str, str]], dataframe: pd.DataFrame, max_history_tokens: int = 1000) -> str: This function combines the DataFrame along with the conversation history into one string and trims the size down to be sent to the Gemini model. This is what allowed the model to interpret the context of the conversation thread and data before giving a response.

Due to the different aspects of the project, I had to call for a few more functions. I called for def build_dataframe_context(dataframe: pd.DataFrame) -> str: to convert the pandas dataframe into a string format that could be accessed by the Gemini model.then called for def trim_to_token_limit(text: str, max_tokens: int, model_name: str = 'gpt-3.5-turbo') -> str: which trims the text to fit into a token limit.

After this code cell, I used the empty set of history[] and a while loop to store each question after it is asked along with the answers. It displayed a graph of data from the dataset along with the assistant's answer. It delivers analysis and potential questions that can be asked to be answered from the dataset. The limitations of the assistant's abilities were also included in the answer. Another section of the answer shows the next possible steps to take followed by the history of questions/answers asked.

The last cell is able to answer questions using the Gemini model and requires the model to provide step-by-step reasoning. The following was my assistant's step-by-step reasoning based on the question: What is the average funding per student in California?

```
Testing the model with reasoning prompt:
      Reasoning Steps:
      1. **Filter the data:** Identify all rows where the `state` column
is equal to 'California'.
      2. **Extract funding:** From the filtered rows, extract the values
in the `funding_per_student_usd` column.
      3. **Calculate the average:** Compute the average of the extracted
funding values.
      Final Answer: The average funding per student for schools in
California is $12179.56.
```

Overall, this assignment was challenging, but I learned valuable information for moving forward in this field of study. This one project almost knocked my hope down, but at the end seeing it run the way I wanted it to was rewarding. I displayed visuals, used external sources, provided sample questions throughout the project to show the differences in each cell, and learned the power of perseverance in this field. I plan on practicing on more of these types of projects in order to gain more skills for future job roles.