Project Solution Overview

Our project allows users to ask natural language to ask questions and receive answers about movies. The data is restricted to titles included in the IMDB movie dataset from Kaggle. Our solution also allows you to compare the results of our search redsults to the results that you might get if you issued the same query to IMDB through their search functionality, or using Google search. Usage instructions are included in the user interface of the solution itself.

How It Works

Our solution runs as a web application, using a locally deployed Flask web server. When the Flask server starts up, it reads several datasets into memory from a SQLite database, saving movie data as Pandas dataframes. Finally, the Flask app initializes an integration with GPT-3 using the datasloth package.

From a simple HTML page, a user can use the search control to type a natural language query and click "Submit". After submitting the query, a few things happen sequentially:

- The request is passed to the `sloth.query()` API, taking the context of the in-memory Pandas dataframes
    - `sloth` generates the request to GPT-3's `text-davinci-002` natural language model
    - GPT-3 responds with a generated SQL query, based on the text input and the context that was passed
    - `sloth` parses the SQL query queries the relevant Pandas dataframes
    - If the query was valid, the method returns with the data, or provides an error message if it was not valid
- The request is passed to the `google_search()` function, which issues an HTTP request to Google Search and returns the top 10 results
- The request is passed to the `imdb_search_movies()` function, which issues an HTTP request to IMDB and scrapes the web page for the relevant movies included in the search results
- The request is passed to the `imdb_search_people()` function, which issues an HTTP request to IMDB and scrapes the web page for the relevant actors/actresses included in the search results

Once all of the results are processed, the HTML results are rendered and the user is presented with the search results.

Setup and Usage Instructions

The solution requires the following runtime dependencies, which must be included in the default `python3` runtime environment to successfully build the Flask server and run the code. Note that we have tested our solution using a Python 3.9.6 runtime, and make no guarantees that another version will offer the expected behavior.

Run the following from a bash shell to activate your environment and install the required dependencies:

```
git clone https://github.com/hainesmichaelc/CS410FinalProject.git
conda create --name envTeam-MK python=3.9.6
conda activate envTeam-MK
pip install flask google urllib3 bs4 pandas datasloth
```

Next, generate an OpenAI API key and copy it to the clipboard. Paste it in the `config.py` file, replacing `"sk_secret"`, so that it can be used at runtime.

If you are starting the application for the first time, you need to initialize the SQLite database. You can do this by opening the `./scripts` directory in a bash shell and running the `python ./init_db.py` script (this might take ~30 seconds to complete).

After initializing the database, start the Flask server from the root project directory (containing `app.py`) by running `./scripts/start_app.sh` from a bash shell.

Access the application by navigating to `http://localhost:81/`. Further usage instructions are documented in the user interface. Try out the queries that we initially ear-marked as our validation set:

- What other movies was the actor that played Jack Sparrow in?
- Who is the girl from Ex Machina and what else is she in?
- What movies are the villain from Thor Love and Thunder in?
- Who directed the best movie of 2016?
- What is the best movie with Rachel from Friends?
- What are top RomComs / Dramas that Adam Sandler starred in?
- When did Hagrid die, what was cause of death, and what other movies was he in?

Note that the latest movie in the dataset was released in 2017, so you will not have luck asking questions about movies that have been released since then. Happy searching!

Team Member Credits

- **Kevin Eveker**
  - Primarily focused on UI, web scraping Google and IMDB search results, and initially setting up the Flask server
- **Michael Haines**
  - Primarily focused on Pandas data wrangling, SQLite database creation, researching/testing GPT-3 integration, and solution hardening