Intelligent Dining Decision Assistant Application

Group Members:

Pauline Brunet, NetID: pbrunet2 (Captain)

Catherine Orlando, NetID: co24

Austin Harmon, NetID: austin31

Mitchell Kopczyk, NetID: kopczyk2

Louis Hamilton, NetID: louisch3

Topic Chosen:

Building an application that aids indecisive users in making informed dining choices by providing personalized restaurant recommendations and insights.

Problem and Relevance to Theme and Class:

Choosing a restaurant can be a complex decision-making process, made even more challenging due to the vast amount of online information available. Our application intends to make this process more manageable and intelligent. The chosen topic aligns with our Text Information Systems class by incorporating concepts such as semantic analysis, relevance, probabilistic ranking, and collaborative filtering to improve the accuracy and relevance of restaurant recommendations and insights.

Datasets, Algorithms, or Techniques Planned:

Datasets: Utilize publicly available restaurant APIs for obtaining detailed information, reviews, and ratings.

Algorithms: Our current algorithm for relevance (R) is R = (Number of unique query terms in document / Number of query terms) * 100

Techniques and Concepts from the Course:

Semantic Analysis: Understanding user queries and reviews to generate relevant restaurant recommendations.

Bag of Words Representation & Vector Space Model: For text representation and understanding the context of user preferences and restaurant information.

Probabilistic Relevance Ranking for Text Retrieval: Ensuring that the most suitable restaurant recommendations are shown to the users.

Collaborative Filtering: Using user-item interactions for personalizing restaurant recommendations.

Evaluation Metrics (Precision, Recall, MAP): To evaluate the performance and relevance of our recommendations.

Demonstration and Programming Language:

We plan to conduct user-based evaluations to receive feedback on the relevance and usefulness of our recommendations, as well as the overall user experience of our application.

Programming Language:

We intend to use languages such as Python for backend development and potentially JavaScript along with suitable frameworks for frontend development.

Application Outcome:

Every time a user enters a query and clicks the "Find Restaurants" button, the system will generate a Python list that consists of a collection of restaurants that are relevant to the query. The generation of this list is made possible by an inverted index. Next to each restaurant in the results, the city, state, and a relevance score will be displayed. The relevance score is calculated based on the following formula:

Relevance = (Number of unique query terms in document / Number of query terms) * 100

The only exception to the relevance formula is when certain stop words are entered, such as "and" and "the". Stop words are ignored during the relevance calculation.

In this scenario, each restaurant name displayed in the results represents a document. Each document is a text file that contains the name of the restaurant on the first line. The city and state are included on the second line. The next lines contain information about the restaurant, which is mostly the food items offered by the restaurant. When the program is not running, more restaurant text files can be added to the main folder and the program will incorporate these files into the inverted index when it starts.

There is no limit to the number of queries a user can enter. While the program is actively running, the user can continually keep entering queries and all results will be maintained. Each new result list will be displayed at the top of the previous results. If the collection of result lists exceeds the window height, the user will be able to scroll down to the bottom of the results.

The user has the option to filter the results by specifying their location with a city and state name. If the user decides to enter their location, the program will still return the original results, except the results that correspond to a different location will be replaced with the text “Location Filter Applied”.

The location filter will only allow the user to see the results that correspond to the specified location. The user will still be able to view the count of the other search results for different locations, which could encourage the user to try their query with a different location, or remove the entire filter. To be the most effective, the user should specify their location in the query and the location filter.

Workload Justification:

1. Initial Project Proposal Draft (Completed)

- Conducted research and brainstorming.
- Drafted the initial proposal outlining the project's goals and methodology.
- Total hours spent: 10 hours (2 hours/student)

2. Project Proposal Edit and Proofread

- Revised, refined, and proofread the proposal for clarity.
- Ensured that the proposal met all project requirements and guidelines.
- Total hours spent: 5 hours (1 hour/student)

3. First Program Draft Development (Completed)

- Developed the initial version of the program, focusing on core functionalities.
- Conducted basic testing and debugging.
- Total hours spent: 30 hours (6 hours/student)

4. Second Program Draft Development

- Improved and expanded upon the initial code.
- Conducted further testing, identifying areas for improvement and optimization.
- Total hours spent: 25 hours (5 hours/student)

5. Final Program Draft Development

- Conducted final refinements to optimize the code.
- Carried out comprehensive testing and debugging to ensure functionality and reliability.
- Total hours spent: 20 hours (4 hours/student)

6. Initial Progress Report

- Compiled and documented the project's progress, identifying completed tasks.
- Total hours spent: 2 hours (0.4 hours/student)

7. Progress Report Edit and Proofread

- Reviewed and revised the progress report, ensuring accuracy and clarity.
- Total hours spent: 2 hours (0.4 hours/student)

8. Program Documentation Development

- Developed detailed documentation explaining the program’s code and functionalities.
- Ensured that the documentation is clear and understandable.
- Total hours spent: 3 hours (0.6 hours/student)

9. Program Presentation

- Prepared a cohesive presentation to showcase the project’s development process and final product.
- Practiced the presentation to ensure smooth delivery.
- Total hours spent: 3 hours (0.6 hours/student)

Total Workload: 100 hours