CIT 594 Final Project CocktailPedia

Team Algoholic

Team Members

Name	Email
Ruichen Zhang	ruichenz@seas.upenn.edu
Rui Zhang	ruizhan@seas.upenn.edu
Wenrui Zhang	wrzhang@sas.upenn.edu
Yuxin Hu	yuxinhu@seas.upenn.edu

Introduction

CocktailPedia is a Java-based cocktail recommender system. It uses cocktail datasets on the web as the data source to provide recipe lookup and recommendations for users. It also allows users to customize their own recipes, and export them in a text file format.

Project Features

1. Recipe Lookup
2. Recommendation
    - 2.1 Recommend by Classic
    - 2.2 Recommend by Popularity
    - 2.3 Recommend by Preference
    - 2.4 Recommend by Preference & Popularity
3. Discount Combo
4. Recipe Customization

Project Demo

Languages, Data Structures and Algorithms

This project is **Java**-based, but we will use very limited **Python** to assist us in cleaning the dataset. Our project involves the use of data structures such as **graph, tree, map, set, array**.

For example, we will use a **hash table** to store cocktail recipes imported from the dataset, and a **sorted map** to keep track of the popularity of cocktail recipes. We will also maintain a **Trie tree** to enhance in cocktail lookup, so that even if users make a typo or enter incomplete words, they can still get the correct result. To implement the discount combo feature, we will maintain a **graph** to store the drinks on discount as vertices, and discount ratio as weighted edges.

In terms of algorithms, we will run **breadth-first search (BFS)** when searching for matching recipes. We will recommend combos with the best discount to users by running shortest path algorithms, namely **Dijkstra's Algorithm**, to find the shortest sum paths.

Interface Design

1. public Map<String, Cocktail> **loadDataset**(String path);
2. public Map<String, Integer> **initializePopularity**();
3. public Map<String, List<Cocktail>> **buildIndexByPreference**();
4. public List<String> **queryByDrink**(String drink);
5. public List<String> **recommendByClassic**();
6. public List<String> **recommendByPopularity**();
7. public List<String> **recommendByPreference**(String taste);
8. public List<Integer> **recommendByDijkstra**(int source, int target);
9. public Double **priceOfDijkstra**(int source, int target);
10. public List<String> **recommend**(String taste, int option);
11. public boolean **customizeRecipe**(String username, String drink, List<String> ingredients, String style, String path);

loadDataset

loadDataset
Method: public Map<String, Cocktail> loadDataset(String path); **Description:** Load the csv format cocktail dataset, and store the cocktail recipes using a map. **Parameter:** path: file path of the dataset. **Returns:** A map storing cocktails recipes, where the key is the name of the drink, the value is the recipe of the cocktail.

initializePopularity

initializePopularity
Method: public Map<String, Integer> initializePopularity(); **Description:** Use the number of queries of the drink to represent popularity. Initialize the popularity map of all drinks with 0s. **Parameter:** None. **Returns:** A sorted map storing cocktail popularity, where the key is the name of the drink, the value is the number of queries of the drink (0 at first).

buildIndexByPreference

buildIndexByPreference
Method: public Map<String, List<Cocktail>> buildIndexByPreference(); **Description:** Group cocktail drinks by taste preference, and store into a new preference map. **Parameter:** None. **Returns:** A map storing different categories of cocktail tastes, where the key is the taste, the value is the list of cocktails of that taste.

queryByDrink

queryByDrink
Method: public List<String> queryByDrink(String drink); **Description:** Return the recipe of the drink name the user queries for, and update the query count of this drink in the popularity map. **Parameter:** drink: the name of the drink. **Returns:** A list of the drink names.

recommendByClassic

recommendByClassic
Method: public List<String> recommendByClassic(); **Description:** Recommend classic cocktails to users, and return the name of the recommended drink. **Parameter:** None. **Returns:** A list of the drink names.

recommendByPopularity

recommendByPopularity
Method: public List<String> recommendByPopularity(); **Description:** Recommend cocktails with top query counts in the popularity map to users, and return the name of the recommended drink. **Parameter:** None. **Returns:** A list of the drink names.

recommendByPreference

recommendByPreference
Method: public List<String> recommendByPreference(String taste); **Description:** Recommend cocktails with the users' preferred taste in the preference map to users, and return the name of the recommended drink. **Parameter:** taste: the users' preferred taste. **Returns:** A list of the drink names.

recommendByDijkstra

recommendByDijkstra
Method: public List<Integer> recommendByDijkstra(int source, int target); **Description:** Recommend cocktails by using Dijkstra based on constructed graph, and return a list of vertices which represent the shortest path. **Parameter:** source: the source drink. target: the target drink. **Returns:** A list of vertices which represent the shortest path, first node is source and last is target.

priceOfDijkstra

priceOfDijkstra
Method: public Double priceOfDijkstra(int source, int target); **Description:** Recommend cocktails by running Dijkstra, and return the price of all drinks recommended after applying the discounts. **Parameter:** source: the source drink. target: the target drink. **Returns:** The price of all drinks recommended after applying the discounts.

recommend

recommend
Method: public List<String> recommend(String taste, int option); **Description:** Return the recommended cocktail recipe to users based on users' options: Option 1: Recommend by classic Option 2: Recommend by popularity Option 3: Recommend by preference Option 4: Recommend by preference and popularity **Parameter:** taste: the users' preferred taste. option: the recommendation option the user chooses. **Returns:** A list of the drink names.

customizeRecipe

customizeRecipe

Method:

```
public boolean customizeRecipe(String username,
                               String drink,
                               List<String> ingredients,
                               String style,
                               String path);
```

Description:

Allow a user to create a customized cocktail recipe. The user can choose from a list of options we provide, including type of the liquor/spirit, taste, preparation style. Then we assemble the recipe based on the choices made by the user, generate a txt file under the directory named by the username, and store it into a user map.

Parameter:

username: name of the user.
drink: name of drink entered by user.
ingredients: ingredients of drink entered by user.
style: style of drink entered by user.
path: file path of storing the recipe.

Returns:

A boolean value indicating whether the recipe has been successfully saved.

Individual Member Contributions

Feature	Story Point	Assignee
Project design & documentation	4	Ruichen
Data cleaning	2	Ruichen
Load dataset and support user queries	2	Yuxin
Build popularity map	1	Yuxin
Build index preference	1	Yuxin
Recommend by a single feature	3	Rui
Recommend by multiple features	2	Rui
Customize recipe	2	Wenrui
Recommender main class	3	Wenrui
Discount combo	4	All
Testing	4	All

GitHub Demo

[Project GitHub Link]

Any Questions?