Cocktail Recommender System

USING A CASE BASED REASONING APPROACH TO DESIGN A COCKTAIL RECOMMENDER SYSTEM.

SHOBHIT JAIPURKAR – (A0163331R)

VIGNESH SRINIVASAN – (A0163246H)

NAITIK SHUKLA – (A0163426H)

NANDHA KUMAR MURUGESAN – (A0163346E)

M.TECH, KNOWLEDGE ENGINEERING | INSTITUTE OF SYSTEM SCIENCES, NUS

Introduction:

Most recommendation systems work on the principle of asking a bunch of questions to the user and using an elimination based approach to reach a final decision. However, this is problematic as, for a personalized result, one would have to excessively train the machine, or will have to provide multiple inputs to the system. This is not ideal in the real world scenario as that would be tedious.

The primary aim of this assignment is to develop a cocktail recommender system for users based on the ingredients they specify. This is aimed to provide a personalized solution to the user, over a generic rule-based system.

Problem Solving Paradigm:

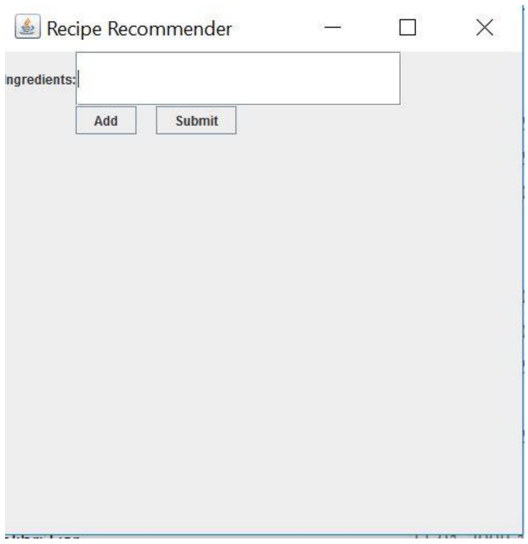
The main problem we're trying to solve here is the personalized response for every user based on their input ingredients. We will solve this using a Case Based Reasoning and there will be a database of approved cases, from which we may be able to select the ideal response for the user.

This approach tries to mimic the human expert approach to the problem as the Case Based solution is modelled on finding a case based on the most similarity, and refining it to fit the users query. We are using Java with and an additional package called jcolibri.

System Design:

- Operational Context:

The Cocktail Recommendation System is based on the requirement of the user's drink preferences. The User Interface has just two buttons, one for adding the ingredient and one for submitting your current list of ingredients. One assumption we are making in the design of the system is that, the Relative Importance of Feature and the Overall Importance of Feature is 2.0.



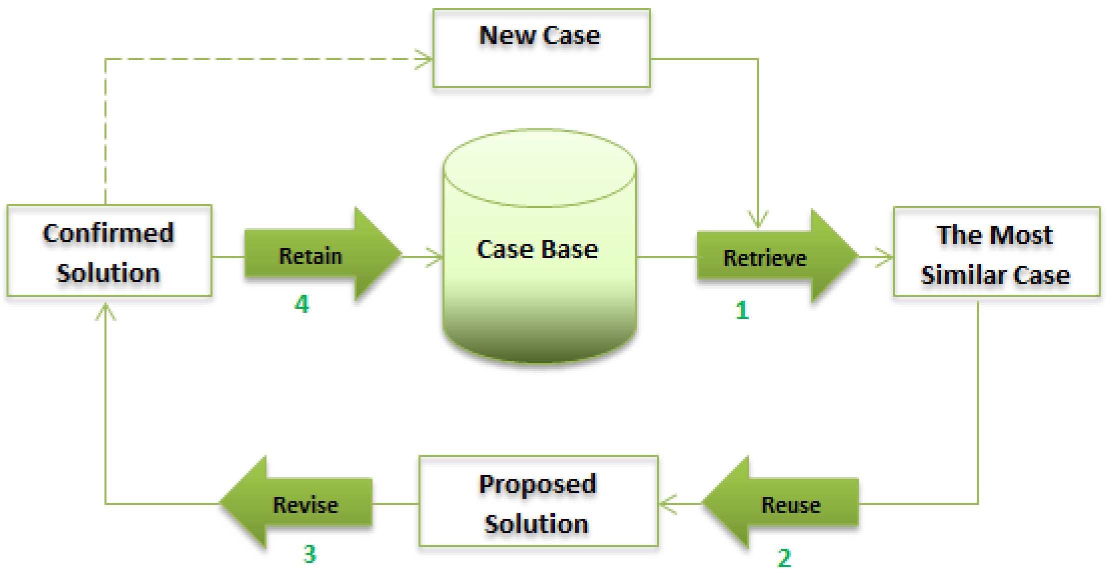
- Functional Description:

The Cocktail Recommendation System receives inputs from the user and outputs a recipe based on the case which is most similar. The main functions taking place here are:

1. Case Retrieval.
2. Case Reuse.
3. Case Revise.
4. Case Retain.

- System Design Architecture:

Here is a high level Case Based System Architecture. The functions are marked from 1-4 and the flow shown here, describes how the software flow works.



- Case-Based Structure and Representation:

Every case is described using an Attribute-Value Representation. The variables in this representation include – (CaseID, Cocktail, Ingredients, Preparation Steps).

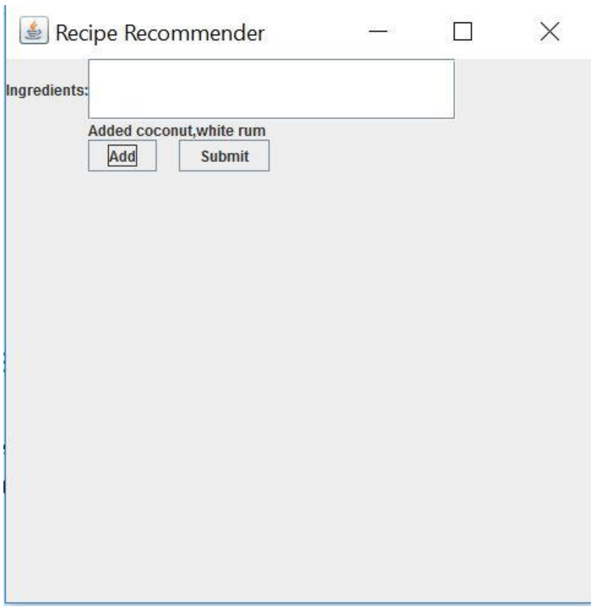
- The CaseID is a Numeric Value.
- The Cocktail name is a Character String.
- Ingredients are comma separated values that represent the ingredients the user wants in his drinks. The quantity and unit data is not stored as the algorithm works mainly on a textual match based on the ingredient list.
- Preparation Steps is a Text Based variable that returns the steps involved in preparing the drink that has been suggested.

Validation and Verification:

To validate the Case Based Recommendation Algorithm, we shall use the test case shown below. Here, we feed the test cases' ingredients to the system which then gives us a result of the top three matches for the cocktail containing those items.

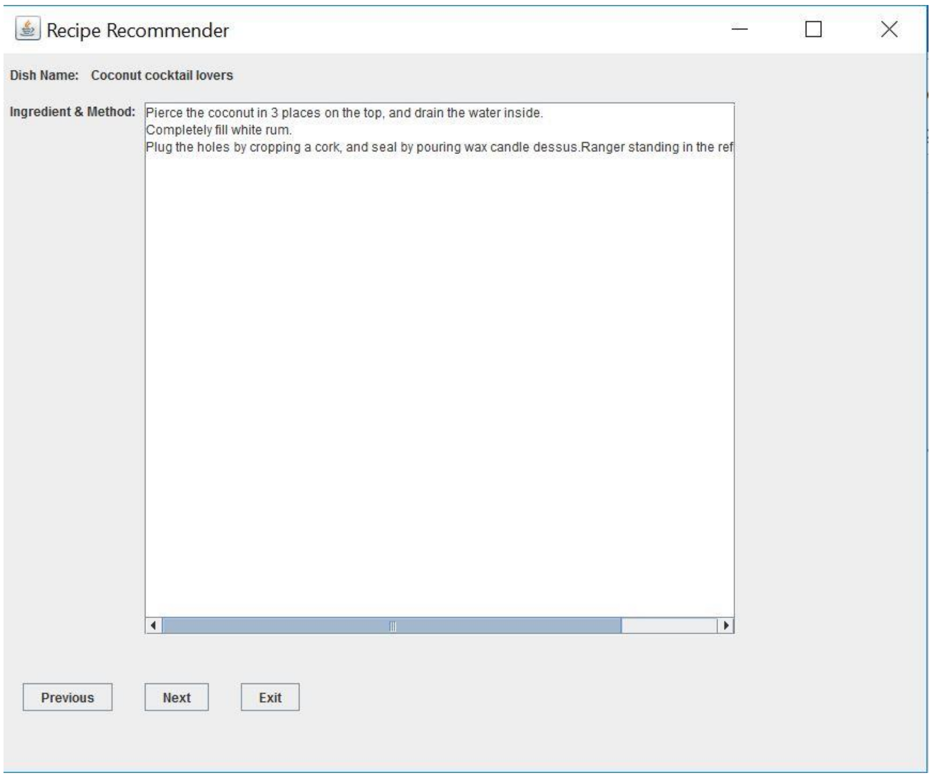
Test Case 1: Coconut and White Rum.

1. We first add the ingredients to the system arguments using the Add button.



Added coconut,white rum

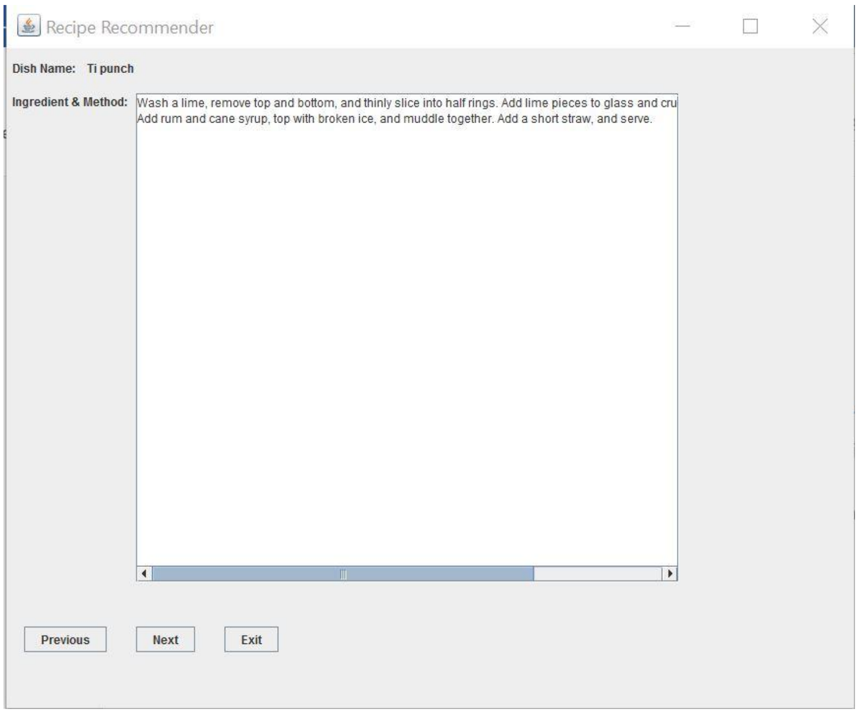
2. After choosing all the ingredients necessary, we hit the Submit button that begins the recommendation process.
3. We are then shown the top 3 results that are obtained for the input, based on their similarity scores to the best case match.



Dish Name: Coconut cocktail lovers

Ingredient & Method:
```
Pierce the coconut in 3 places on the top, and drain the water inside.
Completely fill white rum.
Plug the holes by cropping a cork, and seal by pouring wax candle dessus.Ranger standing in the ref
```

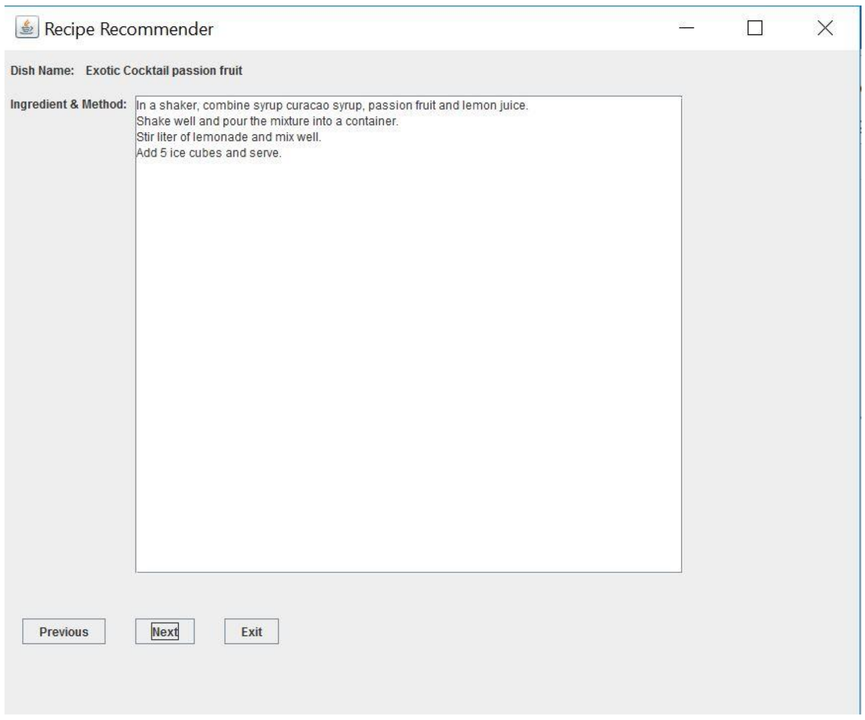
Recipe 1: Coconut Cocktail lovers.



Dish Name: Ti punch

Ingredient & Method: Wash a lime, remove top and bottom, and thinly slice into half rings. Add lime pieces to glass and cru
Add rum and cane syrup, top with broken ice, and muddle together. Add a short straw, and serve.

Recipe 2: Ti Punch.



Dish Name: Exotic Cocktail passion fruit

Ingredient & Method: In a shaker, combine syrup curacao syrup, passion fruit and lemon juice.
Shake well and pour the mixture into a container.
Stir liter of lemonade and mix well.
Add 5 ice cubes and serve.

Recipe 3: Exotic Cocktail Passion Fruit.

Findings and Recommendations:

From the test case above, we can summarize our findings and recommendations on the case based approach to a cocktail recommendation system as follows:

1. The first case is usually the best case match and the ones that follow have a significantly lower similarity value. This explains the large degree of variance between the results from the test case.

2. The system is not wary of spelling mistakes. Since user input undergoes a textual matching with the list of ingredients, a small error in spelling means that the system disregards that input.
3. A recommendation system must start its job right from the input stage. Further iterations to such systems should intuitively suggest “white rum” as a popular option whenever the user inputs “coconut”.
4. As a new user, I will not be acquainted with the list of ingredients. Hence there could be a menu kind of interface where the user can just select the desired ingredients to make the cocktail.

However, most of the changes are UI-based and can be iterated upon easily. The rule engine and retrieval algorithm is found to work flawlessly for all the ingredients.

Conclusion:

Hence, we successfully implemented the design of a Cocktail Recommender System using a Case Based Reasoning approach, with Java.

From the assignment, we also learned the challenges one faces in setting up a Case Based Reasoning System capable of operating on an enterprise scale. We see that, while the CBR approach works best in systems where there is a lot of variance between the input factors, it is also very hard to implement due to the need for a large case base for accurate results.

General Discussion:

As we had discussed earlier, we can conclude that a Case Based Reasoning System is only as good as the case base at its disposal. In some cases, it may find a totally new solution that the human would not have thought of, while in some cases it may keep looping with the same solution.

However, it is not very simple to populate a case base. Especially in situations such as Helpdesks and Customer Care, where the solutions are largely technical in nature, there is the need for a Subject Matter Expert as well. The lack of a subject matter expert could lead to several problems in this approach.

For example, in the case of the Cocktail Recommender system, the user may unknowingly input the ingredients- Citrus and Milk. A subject matter expert or any human will know that milk and citrus cannot be mixed. However, the algorithm will not fire that interrupt and will compute the similarity and produce a result for the same. Here is where the intervention of a subject matter expert would come in use.

In addition to just building a recommendation off immediate values input by the user, we could use this recommendation system to build a history and a profile of the user. By doing this, we could offer personalized suggestions based on previous orders and also recommend some places the user could visit for the cocktail of his choice. The possibilities of a recommender system like this is endless, provided the accuracy and precision is spot on.