Mixology Application: A Case-based reasoning system for Computer Cooking Contest – Mixology Challenge

Rohit Katiyar[1], Sindhuja Sridharan[1], Aditi Goel[1], Nikhil Hegde[1]

University of North Carolina at Charlotte, North Carolina

Abstract: In today's world computer programs are used for a variety of purposes. Suggesting recipes with desired and undesired ingredients by the computer is an interesting and exciting field in AI. In this paper, we have developed a case based reasoning approach to provide a recipe with desired and undesired ingredients selected by the user for the mixology challenge.

Keywords: Computer cooking contest, case based reasoning, mixology challenge, machine learning

1. Introduction

Our application is a Case-based reasoning system which provides a recipe by querying from the existing dataset. If no match is found, then the best possible match of recipes is retrieved and adapted to match user criteria. The new recipe is adapted to use an interesting combination of ingredients while giving the user a good taste. While adapting, the computer system uses ingredients to recipes that a human cook would never possibly think of using and that makes the recipe different and manages us to explore some different, exciting and new tasty flavors. Our application is designed and implemented to provide the adapted new recipe for the Mixology challenge in the Computer Cooking contest.

Artificial Intelligence research was founded in 1956 at a workshop held at Dartmouth College in the aim to develop robots/machines that were as intelligent as a human being. Since then, there has been continuous research in this field to invent intelligent agents. One such agent is the Intelligent cooking agent which can adapt to the limited ingredients available to us at a given time. There are many limitations and problems in developing such agents. To overcome these limitations, the agent must have a capability to learn and adapt from previous data/experience.

Case-based reasoning is one such problem-solving approach which can be used to solve problems by using the existing solutions and adapting these existing solutions to new problems. Case based reasoning overcomes the problems of rule-based expert systems. Our application uses the advantages of existing set of recipes from wikitaable and adapts using a Score approach to create a new recipe as per user preference.

2. Design

Below is the framework of the mixology application and its workflow:

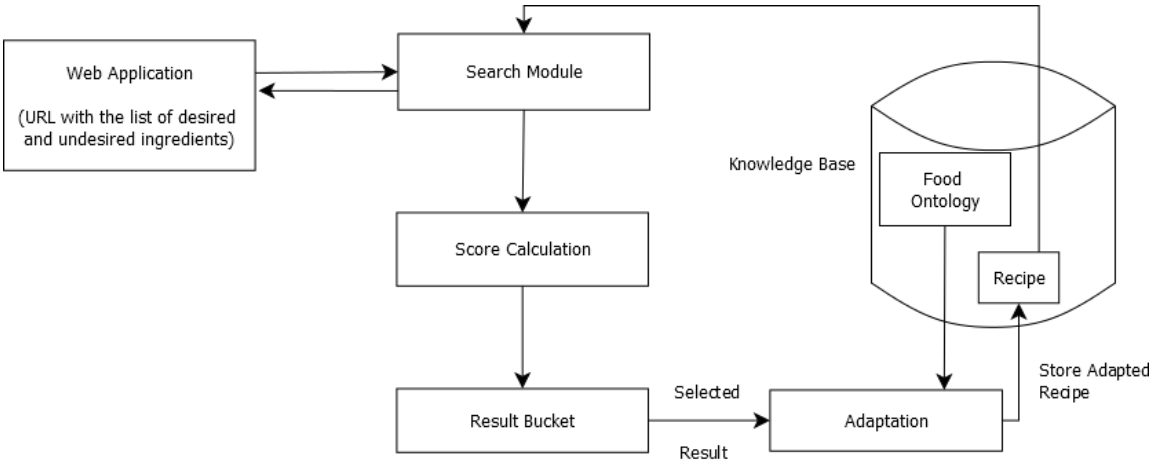

Figure 1: System Design/Workflow

The system takes the input from the web application. The URL consists of the list of desired and undesired ingredients. These lists of ingredients are passed to the search module, which uses them for multiple phases of search to find out the list of relevant recipes. Score is calculated for each of the recipes in the result and the recipes are sorted in descending order based on the score. These recipes are stored in the result bucket.

The top recipe from the bucket is picked, analysed and adapted. Then it is sent as the result recipe to the user. If the user gives good rating(>0) to the recipe then the recipe is retained in the database for the future use else it is discarded and second recipe is selected from the result bucket for the same procedure.

The knowledge base of the system is in two parts: Food Ontology and Recipe. Food Ontology contains the knowledge about food like their name, type, base name ex. Strawberry is a berry same as Raspberry. The Recipe part contains all the recipes in the cocktail section of WikiTaaable.

3. System's CBR cycle

Case-based reasoning has 4 phases. Below is the description of the CBR phases as per the design and development of our mixology application.

3.1. Retrieve

As soon as the system is queried for the recipe with the list of desired and undesired ingredients, it performs multiple phases of search operation in the database for the retrieval of the right recipe. The first retrieval is a simple search containing all the desired and undesired ingredients. The second search tries to fetch the recipes which have all the desired ingredients ignoring the undesired ingredients.

If the first and the second search returns null set, then we generalized the ingredients i.e. replaced the ingredients with their base name like White rum is replaced with rum. After generalizing we again run the search query with more liberal search i.e. even fetching the recipes who have only one desired ingredient or none desired ingredient but all undesired ingredients.

Then we calculated “Score” for each recipe in the result set and sorted them in descending order. The recipes with the highest score (many recipes can have same highest score) are picked for adaptation and testing.

3.2. Reuse

The system keeps on storing the adapted version of recipes as the new recipes if the user gives good rating to the adapted recipe. So, if next time the user query contains the same combination of desired and undesired ingredients, the system will reuse the last stored adapted recipe for the current query.

3.3. Revise

It is based on manual feedback from the user. If the user gives the bad rating i.e -1 or 0 to the result recipe, then the system runs one more adaptation cycle by picking the second-best recipe from the results.

3.4. Retain

If the user gives good rating i.e. greater than 0 to the result recipe then the recipe is stored into the database, if it is the adapted recipe, for future use.

4. SCORE calculation using Machine Learning

As a result of the initial retrieval phase of the CBR cycle, there could be 0, 1 or more recipes returned which matches the user preference. If more than one result is obtained, then we use the Score approach to choose a single recipe. If required, the adaptation method is applied to the chosen recipe.

For the Score approach, we compute the weighted vector of the features to obtain a total score for each of the recipes. The scores are then sorted in descending order and the recipe with the highest score is selected for adaptation and testing.

We created features from the desired and undesired ingredients extracted from the query string. Then we created the feature vector and used it to search for recipes and fill the corresponding vector. If an ingredient (desired or undesired) is present in the recipe, then the feature vector has 1 against that feature ingredient else 0. Then weight vector is calculated for all the searched recipes. Desired ingredient gets the highest weight i.e. +500 and undesired ingredient gets the minimum weight i.e. -500. Then the “Score” is calculated by multiplying the two vectors and adding the values.

For an example suppose the Query String contains Desired Ingredients [Vodka, Rum, Blueberry] and Undesired Ingredients[Gin, Strawberry, Whisky]. Table given below calculates the scores for five recipes with different ingredients based on the query string. The feature vector created from the query string ingredients will contain 6(first 3 features represent desired ingredients and last 3 features represent undesired ingredients) features.

For Recipe 1 the score is calculated as follows:

1x500 + 0x500 + 1x500 + 0x-500 + 0x-500 +0x-500 = 1000.

Recipes	**Ingredients**	**Feature Vector**	**Weight Vector**	**Score**
Recipe 1	Vodka, Blueberry, Raspberry	[1,0,1,0,0,0]	[500,500,500, -500,-500,-500]	1000
Recipe 2	Vodka, Rum, Gin	[1,1,0,1,0,0]	[500,500,500, -500,-500,-500]	500
Recipe 3	Vodka, Rum, Blueberry, Whisky	[1,1,1,0,0,1]	[500,500,500, -500,-500,-500]	1000
Recipe 4	Vodka, Gin, Whisky, Strawberry	[1,0,0,1,1,1]	[500,500,500, -500,-500,-500]	-1000
Recipe 5	Vodka, Raspberry	[1,0,0,0,0,0]	[500,500,500, -500,-500,-500]	500

In the above table the highest score is 1000 and there are 2 recipes with that score. The system will pick both for adaptation and later will select the best one based on user query.

5. Conclusion

The current system can give the right recipe based on the request. However, it does not consider extra choice like gluten-free, on-diet ingredients, neither it calculates the total calories of the result recipe. In the future scope all these factors can be handled in the system. The system's knowledge base can be made more strong and meaningful so that it can handle complex adaptations like if replacing citrus ingredient with any closer non-citrus food.

6. Reference

[1] http://wikitaaable.loria.fr/index.php/Main_Page

[2] Am´elie Cordier, Valmi Dufour-Lussier, Jean Lieber, Emmanuel Nauer, Fadi Badra, Julien Cojan, Emmanuelle Gaillard, Laura Infante-Blanco, Pascal Molli, Amedeo Napoli, et al., Taaable: a Case-Based System for personalized Cooking

[3] Alexandre Hanft, Norman Ihle, and Régis Newo, Refinements for Retrieval and Adaptation of the CookIIS application

[4] Alexandre Hanft, R´egis Newo, Kerstin Bach, Norman Ihle, and Klaus-Dieter Althoff, CookIIS – A Successful Recipe Advisor and Menu Creator

[5] Kerstin Bach, Klaus-Dieter Althoff, Julian Satzky and Julian Kroehl, CookIIS Mobile: A Case-Based Reasoning Recipe Customizer for Android Phones.