Great Snellius Bake-off Logbook

Computational Creativity
Assignment 1: Great Snellius Bake-off
Chang Liu
Olaf Wisselink

Oct 9

- Task:

1. Finish the Simple Piere code tutorial separately
2. Familiarise ourselves with JSON

Oct 12

- Task:

1. Discuss the main parts to work/improve upon based on the tutorial code. We decided to mainly focus on how a recipe is represented in the inspiring set and improving the existing fitness function.
2. Look at and collect some initial recipes.

- Discussion & Result

For the inspiring set all recipes are taken from *allrecipes.com*. Initially we focused only on cookie recipes, to perhaps later include more classes like cake and pie. We started with manually formatting three recipes into a JSON file.
We decided a recipe is represented by its name and the ingredients. We thought about making a hierarchical tree structure with a ‘super’ class module included in the ingredients. We thought this provided helpful information in deciding on a recipe’s fitness based on the relative proportions of different classes of ingredients. An ingredient is now represented by a class, a role, an amount and a unit. The role is the function of the ingredient and mostly provides information about the taste. We initially decided to keep the original units (cup, teaspoon, tablespoon) from the recipe and to think of a way to convert these to a uniform unit later.

```
"ingredients": [
        {“base”:[
                {"ingredient": "flour", "role": "savoury", "amount": 1, "unit": "cup"},
                {"ingredient": "butter", "role": "savoury", "amount": 0.5, "unit": "cup"}
        ]}

        {“binding”:[
                {"ingredient": "egg" , "amount": 2, "unit": "eggs"}
        ]}
        {“rising”:[
                {"ingredient": "baking soda", "amount": 0.5, "unit": "tsp"}
        ]}
        {“flavour”:[
                {"ingredient": "salt",  "role": "salt", "amount": 2.75, "unit": "tsp"},
                {"ingredient": "vanilla extract", "role": "sweet", "amount": 1, "unit": "tsp"}
        ]}
]
```

Oct 14

- ## Task:

1. Further work on the representation of a recipe in the inspiring set
2. Think about how to determine a recipe’s fitness.

- ## Discussion & Result

1. The newest version of our inspiring set is as follows. We abandoned the multi-layer format and gave each ingredient a class parameter. This format made it more easy to integrate into the mutation, and selection and crossover process.

```
{
  "name": "Salted Chocolate",

  "ingredients": [

    {"ingredient": "flour", "class": "base", "role": "savoury", "amount": 1, "unit": "cup"},
    {"ingredient": "butter", "class": "base", "role": "savoury", "amount": 0.5, "unit": "cup"},

    {"ingredient": "egg", "class": "binding", "amount": 2, "unit": "eggs"},
    {"ingredient": "baking soda", "class": "rising", "amount": 0.5, "unit": "tsp"},

    {"ingredient": "brown sugar", "class": "flavour", "role": "sweet", "amount": 7, "unit": "tbsp"},
    {"ingredient": "salt", "class": "flavour", "role": "salt", "amount": 2.75, "unit": "tsp"},
    {"ingredient": "vanilla extract", "class": "flavour", "role": "sweet", "amount": 1, "unit": "tsp"},

  ],
},
```

2. We thought of ways to meaningfully measure a recipes’ fitness. We came up with two recipe parameters, namely 'similarity’ and ‘novelty’. Similarity gives us information about a recipe in terms of its similarity to all recipes in the inspiring set. We assume that a recipe’s quality is higher when it shares aspects commonly found in the

inspiring set. Novelty is the opposite of similarity. We assume a recipe is also of good quality and novel if it has an aspect that is not commonly found in the inspiring set. We think that because the two parameters oppose each other, they will provide a balanced mix of both similarity and novelty in a recipe.

Oct 16

- ## Task:

1. Normalising the recipes
2. Optimize the inspiring set and get more recipes.
3. Start implementing the fitness function

- ## Discussion & Result

1. We converted all the original ingredient units (cup,tsp,tbsp) to millileters. This is possible because the conversion is independent of the weight of the ingredient. We did this because it provided a way to easily compare and mutate amounts of different ingredients. Also, we can now normalise the recipes to how much servings the recipe yields. For the presentation part we can again convert back to the original units, because they are more common and readable.

2. We added the entries “yield”, “star rating”, “calories”, and “number of reviews”. This information was all included in the original recipe. Yield is the amount of cookies/servings, and the star rating is the amount of stars users from the website rated the recipe. We initially wanted to use the ratings, calories and reviews as additional parameters of measuring fitness, but we ended up not using these parameters so far as it turned out it not to provide more meaningful information in the calculation of fitness. We expand on this more in the last section, future work. Additionally we manually added more recipes to the inspiring set. We now have 6 recipes.


```
{
  "name": "Salted Chocolate",
  "yield": 12,

  "ingredients": [
     ...
  ],

  "star rating": 4,
  "calories": 236.4,
  "number of reviews": 9
},
```

3. We started working on the fitness function by implementing the similarity measure. At the beginning of the program we calculate the amount of unique ingredients and number of appearances in the inspiring set. Then in the fitness function a recipe's similarity measure is a weighted sum of all its ingredients, represented by a number between 0 and 1. Ingredients that are more common in the inspiring set weigh more.

Oct 17

● Task:

1. Improve the mutation function:
    a. Make the mutation on "amount" less random
    b. make the mutation on "ingredient name" more reasonable with regard to our own system
2. Further work on the fitness function.

● Discussion & Result

1. We got the inspiration from Morris' paper[1] on how to mutate the amount of a single ingredient more in a more informed way: we calculate the mean and the standard deviation to build a normal distribution profile for each of unique ingredients. Each time when the mutation happens on the amount of a randomly selected ingredient, we randomly choose a number from that ingredient's own distribution profile to substitute the original number. This would make the mutation process more informed as the newly updated value would always fall in a reasonable range (we assume the recipes in the inspiring set are all good recipes so that the amount of each ingredient is reasonable, thus the distribution profile controls the newly generated number to be always reasonable).
   The second change we made in the mutation function is the mutation on the single ingredient's name. In the original source code, there are fewer terms for each ingredient *(ingredient_name, amount, unit)*. In our version the terms to describe each ingredient is more complicated *(ingredient_name, class, amount, unit)*, and the unit is not yet unified (we only unify the "amount" at the beginning, while keeping the unit unchanged for future reverse operation). So instead of randomly choosing a name from the inspiring set to substitute the original ingredient name, we decide to produce a random number as the index to locate the newly selected ingredient. So each time when the mutation happens on the "ingredient name", we would grab all the related description information *(ingredient_name, class, unit)*, but not the amount to update the original information to make sure the consistency.

[1] Morris, Richard G., et al. "Soup Over Bean of Pure Joy: Culinary Ruminations of an Artificial Chef." *ICCC*. 2012.

2. We implemented an additional measure in the fitness function: the ‘class ratio’. For each class of ingredients in a recipe we compute its relative size. We then compare this to the average size of that class of all recipes in the inspiring set. This class ratio measure tells us if the different classes of ingredients are in a good proportion to each other. We assume all the recipes within the inspiring set are of good quality, thus the class proportion in the inspiring set is reasonable.
   The class ratio is the sum of all class proportions, represented by a number between 0 and 1. The closer the proportions are to the average proportions, the higher this number. The fitness function now has three measures: similarity, novelty and class ratio.

   This pseudocode example shows the ‘class ratio’ computation for the class flavour for one recipe:
   flavour_weight[recipe] /= total_weight[recipe]
   flavour_weight[recipe] = 1 - abs(flavour_weight[recipe] - avg_flavour_weight)

Oct 18

- ## Task:

1. Work on recipe presentation:
    a. Name generator function
2. Finish the fitness function

- ## Discussion & Result

1. We decided to generate the name for a new recipe based on ingredient names with the highest amount for the classes the “flavour” and “topping”. The “flavour” name and the “topping” are combined as the name for the newly generated recipe. We only need to do this once, after all the final population is filtered out.
2. Because of time and coding issues we did not have time to finish the fitness function and implement the novelty measure. So so far, our fitness function only measures similarity, based on what kind of ingredients and their relative proportions.

Reflection on creativity

Bases on Ritchie's Empirical Criteria for Attributing Creativity, we judge the artefacts produced by our creative system based on:

1. Typicality
   We measure the typicality of the recipe we generated by the **recipe_similarity** measure we defined. So if the newly generated recipe contains more ingredients that are adopted by more recipes in the inspiring set, we consider the recipe to be more reasonable and typical.

2. Novelty
   We measure the typicality of the recipe we generated by the **novelty_score** measure we defined (but not yet implemented). If the recipe is relatively novel, then the combination of ingredients might be rare.

3. Quality
   We measure the typicality of the recipe we generated by the **class_ratio** measure we defined. If a recipe is of good quality, it should have a relatively good combination.

Future work

1. Finish the fitness function. Due to said problems, the novelty measure still needs to be implemented as previously described.
2. Until now the inspiring set is pretty small for our system and contains 6 recipes. These have been added manually, which cost a lot of time. In the future, we might need to find a way to crawl data in batches from the website, as html data follow certain rules. More recipes provide more variety and novel combinations of ingredients, which so far is lacking.
3. We document calories information at present. We could also calculate the total calories a newly generated recipe contains, as long as we build a standard dictionary that documents how many calories each ingredient contains per gram.
4. We could adopt a MLP (multi-layer perceptron) in the fitness value calculation. During the fitness function design, we find it difficult to set a standard on what a perfect cookie should be like as we are not really the domain expert. At present we build the fitness function based on the assumption all the recipes in the inspiring set are all good recipes and we take them as a reference.
   But we find as we get users' star rating, so we could use this as a signal to decide if the recipe is good or not. We could use the star rating as a target/dependent variable to train a multi-layer perceptron which could tell us the quality of a recipe (as it analyzes the potential relationship between independent variables and the target ratings). The independent variables to be sent into training could be the values we just got: (`the recipe class ratios, the recipe_similarity, the recipe novelty score, the calories etc.`)

5. The presentation so far is limited. After the program finishes after *n* runs, we print the best recipe in the population. This recipe has a generated name, a list of ingredients in common, readable units. However, we could improve upon this, by giving cooking instructions, or show images based on the ingredients.