Logbook of Computational Creativity Bitwise Bakeoff

Initial Idea

16.10.2024

Our initial idea for the Bitwise Bakeoff was to create a mochi recipe generator. Mochi is a traditional Japanese food made from glutinous rice, known as mochigome in Japanese. This soft, chewy dough forms a rice cake that can be wrapped around various fillings. There are many variations of mochi, as it can be filled with ingredients like red bean paste or ice cream. Mochi flavours can be both sweet and savoury. Because of this flavour variety, we thought mochi would be the perfect food as an inspiring set for a recipe generator.

We plan to use this and this website for our initial set of inspiring recipes.

Additionally, to generate unique and unusual mochi flavours, we planned to create our own list of ingredients to combine with mochi. To calculate the fitness of each ingredient, from both the external recipe sources and our own list, we used the ratings (1 to 5 stars) from the existing recipes. For each ingredient found in a recipe, we assigned it the recipe's rating. For ingredients from our own list, we rated them ourselves and calculated the average score based on our ratings.

Since mochi is originally from Japan, for the presentation of our recipes we liked to maintain the Japanese style. Therefore, we planned to use a text generator that creates text from Japanese symbols, like 爪ㄖ匚卄丨, and used this font for the titles of the recipes. Another idea is to present the recipes in a kawaii style. Kawaii is a part of Japanese culture that emphasizes cuteness. We can achieve this by creating AI-generated images of cartoonish cute mochi designs that match the ingredients in the recipe and using soft pastel colours associated with a kawaii drawing style for the cookbook. We wanted to create a tiny pocket cookbook as that would make it instantly cute and it would fit well within our Japanese mochi theme. Moreover, we wanted to use generative AI to create fitting titles for the mochi recipes. We hand-designed the cookbook and used placeholders for the ingredients, instructions and images of the generated recipe.

We aim at increasing the perceived value of the recipes by staying within this Japanese-themed presentation for the Mochi cookbook.

Updates on idea/new goals

23.10.2024

At some point we realised that by adding our list of random ingredients, we would essentially be creating monsterous mochi. This sounded quite nice to us and resulted in the renaming of our project from "Cookie monsters" to "Mochi Monsters". Thus, the title of our cookbook will be Mochi Monsters.

Inspiring set & creation of knowledge base

16.10.2024

We gathered recipes for our inspiring set using two different websites with both around 20 different mochi recipes as our inspiring set. We webscraped the recipes and created our own database with it. We decided that besides existing recipes, we wanted to add some loose ingredients and named this our "weird ingredient list", as we were interested in adding ingredients that you would normally not associate with mochi. For this list, each of us added some unexpected ingredients and rated them on a 1 to 5 scale for the fitness function. Some of the recipes from the two websites we used in our inspiring set also did not have a rating, therefore, we gave them our own rating as well.

17.10.2024

With webscraping in general, we encountered lots of exceptions. So we had to hard code these exceptions. For example. The placement of amount was sometimes in the incorrect html label. The amount of 3/4 cups was not really amount, but rather string based. Or there were references in the ingredients to other recipes instead of single ingredients. So we had to come up with workarounds to overcome these exceptions.

21.10.2024

Additionally, due to allergies, we created a class of ingredients that contains nuts, peanuts, and gluten. To normalise every recipe we standardised every recipe to a serving of 10 mochis and converted every unit to grams. Also with this, there were some exceptions to the available units. For example, sometimes "slices" were used to describe the amount of banana, or "pkg" to describe a package of beans. With available online information, we hard coded this into available units to convert them, such as "oz".

Implementation recipe generator

21.10.2024

We chose to write an algorithm that runs by itself without intervention from a user. The stopping criteria is 500 fixed generations, including steps for crossover, mutation, normalisation and evaluation. It returns the best recipe.

26.10.2024

For the final implementation, we mainly based our algorithm on the algorithm used by the PIERRE assignment. We made some adjustments to the algorithm to make it a better fit for our fitness function. Adjustments were as follows:

- fitness function alterations
- mutation with our "weird ingredient list"
- normalization: instead of sum, average amount and added average rating with duplicates and different scale (500 instead of 1000)

Below, we will futher explain each of the adjustments.

The fitness criteria

16.10.2024

At first, we wanted to use "rating" as our evaluative criterium. We wrote a fitness function where only the best rated ingredients would fit in the recipe. The best recipe rating would have the highest average rating.

21.10.2024

We noticed that by using only the rating, that it ended up generating recipes that used only ingredients with a rating of 5. This meant that most ingredients were automatically excluded, and thus also the possibility of a "surprising" recipe. For that reasing we decided to add a random evaluative criterium (such as "length of ingredient list" in the simple PIERRE example). We settled on using the amount of consonants in an ingredient. We gave weights to each criteria, with rating a weiht of 0,1 and consonants a weight of 0,9. Moreover, the fitness function excludes ingredients containing gluten or nuts by ascribing them with a value of 0.

23.10.2024

After giving the fitness function some thought, we figured that ingredients with long names would automatically be chosen over ingredients with short names. We decided to calculate the average number of consonants in a recipe. Furthermore, what is important to make mochi, is that it contains (one) flour. That is why we incorporated this into the fitness function, where fitness is set to 0 if the recipe contains no flour or more than one flour. Since we are making mochi balls that do not need to be baked, we also set the fitness to 0 when eggs and baking powder were included in the recipe. Finally, the fitness is set to zero when allergens are included. This led to the following fitness function:

- The number of consonants with a weight of 0,9.
- The rating of the recipe with a weight of 0.1.
- The presence of 1 flour (binary),
- The presence of eggs or baking powder (binary).
- The presence of ingredients that are categorised as allergens (|binary).

```
r['fitness'] = (rating*0.9 + cons*0.1) * flour_presence * forbid
```

All binary factors were combined as "forbidden" factors in our fitness function.

26.10.2024

We found out that basing the fitness of the average amount of consonants did not give very inspiring recipes: it still used the ingredient with the longest name. Also, it produced recipes with only that ingredient, as adding others would bring the average amount of consonants down. We added to our fitness function that any difference in a length of 7 ingredients would result in a penalty, lowering the fitness, just to see if it would make a difference. The recipes that came out were better already. However, the consonant criterium limited our ingredients to a small sample, so we removed the consonant criterium from our fitness function.

Moreover, after going through many recipes, we learned that the base ingredient of a mochi is not only flour, but flour in combination with either a liquid or a puree in combination with a liquid. Also, that butter is not used in the dough, only possibly in the filling.

We realised that 7 is maybe a bit of a random number, so we argued that it is nice to make it a range. The minimum amount of ingredients necessary to make mochi is 3 (a flour, a liquid, and a filling). This little ingredients would not spark exciting recipes. We thus chose to set the minimum of ingredients to 5, since we are building a creative system. With too many ingredients, the individual ingredients might lose their flavour. Thus, we settled on a maximum of 8 ingredients. We chose a range of 4 to 8 ingredients. With a length smaller than 3 ingredients, it would not be a mochi, and therefore it would be excluded, and above 8 we included a penalty.

Therefore, we made some adjustments to our fitness function:

- The rating of the recipe
- The presence of min 1 liquid OR puree AND 1 flour (binary)

- The presence of eggs or baking powder (binary)
- The presence of ingredients that are categorised as allergens (binary)
- The presence of max 1 butter and min 1/max 2 liquids (penalty)
- The presence of minimum of 5 ingredients and max 8 ingredients (penalty)

```
r['fitness'] = max(0, rating - length_penalty -
ingredient_constraint)*forbid*dough_presence
```

27.10.2024

We encountered that the pool of ingredients was limited due to including only the highest rating as fitness. We decided to enlarge the pool of ingredients by making the rating scaled from 4.7 - 5 as accepted rating. When the rating falls between this range, a random rating will be chosen between 4.7 and 5 to give all ratings between this an equal change.

Our final fitness function:

- The scaled rating of the recipe
- The presence of min 1 liquid OR puree AND 1 flour (binary)
- The presence of eggs or baking powder (binary)
- The presence of ingredients that are categorised as allergens (binary)
- The presence of max 1 butter and min 1/max 2 liquids (penalty)
- The presence of minimum of 5 ingredients and max 8 ingredients (penalty)

```
r['fitness'] = max(0, scaled_rating - length_penalty -
ingredient_constraint)*forbid*dough_presence
```

Mutations

16.10.2024

As possible mutations we used addition, substitution (with our weird ingredient list) and deletion, and changing the amount of ingredients.

Normalisation

25.10.2024

We had not thought of how we wanted to normalise our recipes. The only possible solution was to try out some existing mochi and base our answer on that. As a result, we decided that we wanted each of our mochi to weigh around 50 grams (as our try-out mochi weighed 35 grams and were a bit scanty to our taste). As we generate recipes for 10 servings we decided that we would normalise it to 500 grams of total ingredient weight.

Additionally, with removing duplicates, we added two rules. The first rule is averaging the amounts instead of summing up the amounts. This is mainly done because we splitted duplicates in the original dataset as well. Flour1, flour2, or flour3 as one ingredient was changed to one ingredient for each. However, we did not divide the amount by the amount of ingredients, but decided to keep the amount for these seperate ingredients the same. When there were duplicates in a newly created recipe, we therefore averaged the amount instead of summing up, to avoid too big numbers of amounts. The second rule is averaging the rating of the ingredients, to get a more fair rating for the total recipe based on duplicates.

Limitations

Some recipes from the inspiring set did not have a rating. We solved this by adding our own rating, based on our own liking of the ingredients. We then took the average of our ratings for the recipes. Another problem with the recipes from the inspring set was the ingredients were measured in different units (e.g., cups, ounces, and grams). To achieve a recipe that takes into account a correct ratio of the ingredients, we converted all the units to grams and ml by using the library Sugarcube. This library already contained the density of the products water, sugar, flour, butter, and salt, therefore these ingredients were automatically converted. For the density of other ingredients, we categorised them in classes, such as liquid for milks and sauces, and gave them a general density based on information online. After catergorising all the ingredients, we added their densities to the library, whereafter the units were converted. On top of that, there were some measuring units that were difficult to convert or normalise, such as "a whole/large (egg)". Moreover, sometimes an ingredient was a flour "to dust the surface". We chose to exclude those ingredients.

Reflection on the question of how to evaluate the creativity of your system

To evaluate our recipe generator, we applied the Lovelace 2.0 Test, in which an artificial system is assessed against a set of constraints that serve as our fitness criteria.

Lovelace 2.0 Test In The Lovelace 2.0 Test, an artificial agent (a) is challenged as follows (Riedl, 2014).

- a must create an artifact o of type t;
- o must conform to a set of constraints C where ci ∊ C is any criterion expressible in natural language;
- a human evaluator h, having chosen t and C, is satisfied that o is a valid instance of t and meets C; and
- a human referee r determines the combination of t and C to not be unrealistic for an average human.

In applying the Lovelace 2.0 Test to our system, we structured our evaluation as follows:

- The system (a) must generate a recipe (o) specifically for mochi (t).
- The recipe (o) is required to meet the following constraints (C) without consideration for aesthetic judgments, and it need only achieve a level that an average unskilled individual could accomplish:
  - The recipe must contain exactly one ingredient from the category ‘flour’.
  - The recipe does not contain eggs or baking powder.
  - The recipe contains a maximum of one ingredient from the category ‘butter’.
  - The recipe must contain a minimum of one ingredient from the category ‘liquids’ OR ‘bindings’.
  - The recipe contains a maximum of two ingredients from the category ‘liquids’.
  - The recipe does not contain gluten, nuts or peanuts.
  - The recipe must consist of 5 to 8 total ingredients.
  - The average rating of ingredients used in the recipe must be between 4.7 and 5 (on a scale from 1 to 5).
- The human evaluators (h) – ourselves, Emma, Linthe, and Sanne – assess the recipe (o) and confirm that it represents a valid recipe to make mochi (t) and complies with all specified constraints (C).
- The human referee (r), a fellow student, must verify that the task of making mochi (t) with the given constraints (C) is realistic for an average person.

Evaluation human evaluators:

According to Riedl (2014), the creativity of the artificial agent can be expressed as the mean number of constraints passed. Therefore, each human evaluator (h), Emma, Linthe, and Sanne, conducted a series of tests, increasing the number of constraints from 1 to 8. Each test required the system to generate a recipe that met the specified constraints. The testing was stopped when the system failed to meet the criteria. The highest number of constraints it successfully fulfilled was documented. The system's creativity score was then calculated as the average number of constraints passed across all evaluators. This score indicates the system's capability to handle increasingly complex requirements.

For this evaluation, all evaluators confirmed that the recipes met the eight essential constraints to create mochi. The system's creativity score was calculated as follows:

(∑_i(n_i))/(|h|) =

(8+8+8)/3 = 8

The system scored 8 out of 8, which demonstrates that the system successfully met all constraints in each test. Therefore, the evaluators collectively agreed that each recipe serves as a representative example of a mochi recipe.

Evaluation human referee:

The human referee (r) ensures that the evaluators don't assign the system an extremely difficult combination of t and C, which even for humans would be hard to achieve. A fellow student stated the following:

''The constraints seem realistic. However, maybe it's a bit hard to understand what is highly rated mochi and what not.''

This indicates that while the constraints are generally appropriate, there is some ambiguity regarding the subjective criteria for what creates a highly-rated recipe. Ingredient ratings can be subjective and depend on personal preferences. Therefore, it is important to establish clear rating criteria. However, it can be argued that highly-rated ingredients reflect a typical human cooking experience, where ingredients are chosen subjectively, often based on individual taste preferences.

Riedl, M. O. (2014). The Lovelace 2.0 Test of Artificial Creativity and Intelligence. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1410.6142