AutoChef: Automated Generation of Cooking Recipes

Hajira Jabeen
Informatik III
University of Bonn,
Bonn, Germany
jabeen@cs.uni-bonn.de

Jonas Weinz
Informatik III
University of Bonn,
Bonn, Germany
s6jowein@uni-bonn.de

Jens Lehmann
Informatik III
University of Bonn,
Bonn, Germany
jens.lehmann@cs.uni-bonn.de

***Abstract*—Cooking is an endeavour unique to humans. It is mainly considered an art requiring culinary intuition acquired through practice. The preparation of food is a complex and subjective process that makes it challenging to determine underlying rules for automation. In this paper, we present AutoChef, the first open-source autonomous recipe generator. AutoChef extracts the data from existing recipes using natural language processing, learns the combination of ingredients, preparation actions and cooking instructions, and autonomously generates the recipes. Furthermore, AutoChef uses Genetic Programming to represent and evolve the recipes. The fitness of recipes is designed to evaluate the combination of ingredients, actions and cooking-processes learned from the existing recipe data. Finally, the resulting recipes are translated back into text format and evaluated by human experts.**

I. Introduction

Food plays an important and diverse role in our life, ranging from the necessity to eat to pleasure, culture, recreation and art. Recipe preparation or culinary practice is an intricate combination of art and science that integrates flavours, textures, nutrients and aromas, and results in a food that is edible, tasty, healthy and presentable. Different cultures enjoy a diverse set of traditional recipes in different parts of the world. These diverse groups of food preparation methods, ingredients and spices make up the cultural cuisines. These cuisines have implicit embedded knowledge of geographical preferences related to health, climate, nutritional needs, taste, texture and aroma. Encoding this information automatically into a machine-understandable format is a challenging task, let alone learning from this information to create new recipes by combining ingredients and spices from different cuisines. Contrary to the abundance of artificial creativity in applications like music, pictures, and graphics, its application in culinary arts has remained limited to recipe recommendations, cuisine detection, or ingredient substitution. There are no known autonomous recipe creators apart from (now closed) IBM ChefWatson [1], This closure can be attributed to the underlying complexity of the problem, and direct-yet-subjective impact of the outcome.

In this paper, we present the initial results of open-source AutoChef [2] that evolves new recipes by integrating ideas from machine learning, natural language processing, evolutionary algorithms and genetic programming. We analyze the cooking actions on ingredients, the intermixing of ingredients, and further use this extracted information to generate recipes automatically. We encode the recipes as genetic programs and evolve them through specialised operators aiming to optimise the proposed fitness criteria. The fitness function is designed to evaluate the combination of ingredients, actions, and cooking methods. An overview of our proposed approach AutoChef is shown in Figure 1 and discussed in detail in Section III. Section II covers related work and Section IV shows the evaluation of the approach. Finally, Section V concludes the work presented in this paper.

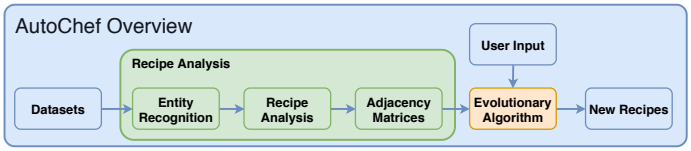


Fig. 1: Overview of AutoChef

II. Related Work

Various lines of research are being conducted for food, cooking, and flavouring. For example, detection of recipes from images [3] [4], recipe recommendation systems [5], and cuisine transformation [6]. In this paper, we briefly cover the automated recipe generation endeavours covering computational arts. One of the prominent efforts in recipe creation was performed by IBM, named "IBM Chef-Watson", and presented in 2014 [1]. IBM built a system that produced novel recipes by introducing new ingredients in existing recipes taken from bon-appetit [7]. The quality of the recipe was determined by novelty measured as the deviation from common recipes, and aroma measured by evaluating the chemical properties of flavour molecules used in the recipe [8]. Chef-Watson often resulted in hard to find ingredients. For unidentified reasons, the project seems to be discontinued and the web reference is no longer functional. The Evolutionary Meal Management Algorithm (EMMA) [9] generates recipes using a machine learning algorithm. The resulting recipes often lack clear instructions and quantities. Erol et al. [10] have developed an approach to discover novel ingredient combinations for the salads. However, they ignore the instructions and the quantities

of ingredients, which is one of the crucial elements of any recipe. EvoChef [11] is one of the prototypes developed for evolving recipes. However, it is constrained by limited data ingestion as well as manual fitness evaluation. A further step in recipe evolution is the semi-automated SmartChef [12], which has successfully performed the semi-automated evolution and created more complex recipes with a partially automated fitness evaluation.

An approach to transforming existing recipes into different cuisine style is presented by Kazama et al. [6]. They create embeddings for the recipe ingredients to make it possible to transform the regional style of a recipe. They used the word2Vec [13] approach to find the embeddings for all ingredients. They measure the distances in the embedding space of different cuisine styles and use these difference vectors to transform recipes into another cuisine style. The information of a particular recipe is only represented by its ingredient list. In comparison, the work of Marin et al. [4] builds recipe embeddings by considering the instructions as well. Their approach uses embeddings generated for each instruction and the ingredient list. They train a neural network to learn a joint embedding of recipes and images that are used to retrieve recipes for given images. The prominent approaches outlined above demonstrate that substantial research is being done in this area. However, we are still far from a fully automated and functional recipe generator.

III. AutoChef: The Recipe Generator

Autochef works in three main steps, as shown in Figure 1. The first step is data curation, in the second step, we perform ingredient detection and recipe analysis to correlate ingredient and cooking instructions. In the last step, this inferred information is used to automatically create and evolve cooking recipes.

A. Datasets

We have selected the following two datasets for the analysis and to extract information about the usage of ingredients and cooking actions

One Million Recipe Dataset The [4] contains one million recipes in English collected from various popular cooking websites. Each item in the data contains a title, an ingredient list, an instruction list and an image of the final product. The recipes are provided in plain text as a JSON array (listing 1)

Yummly Dataset The cooking website Yummly (see also [14]) provides a dataset containing the ingredient list of 39,774 recipes annotated with the regional cuisine style, provided as JSON.

B. Recipe Analysis

AutoChef aims to extract information from existing recipes to learn to generate new recipes. To detect ingredients and instructions, we have used Conditional Random Field (CRF) classifier [15]. We detect cooking actions and ingredients by using part of speech tags generated with the "nltk toolkit" [16] and features like the position in the sentence, word endings and


```
{
  "ingredients": [
    {
      "text": "8 tomatoes, quartered"
    },
    {
      "text": "..."
    }
  ],
  "url": "http://www.foodnetwork.com/\
              recipes/gazpacho1.html",
  "partition": "train",
  "title": "Gazpacho",
  "id": "000035f7ed",
  "instructions": [
    {
      "text": "Add the tomatoes to a food processor
      with a pinch of salt and puree until smooth."
    },
    {
      "text": "..."
    }
  ]
}
```


Listing 1: Example JSON recipe [4]

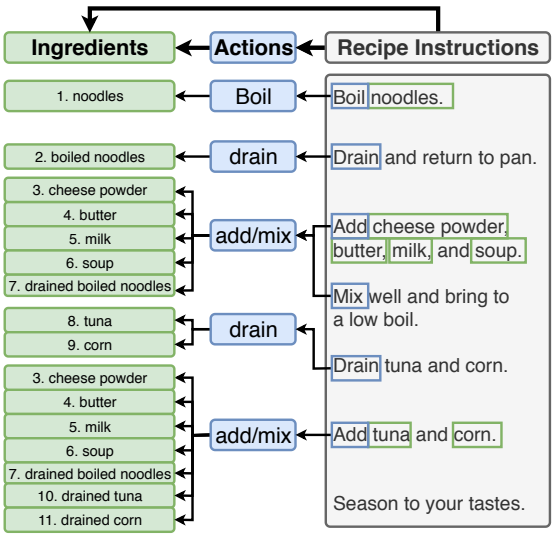


Fig. 2: Information extraction from a recipe

surrounding words. These features are used as input values for CRF training. It turns out that the entity recognition works-better on ingredients than on actions. Therefore, we used a hand-curated list of around 50 fixed cooking actions and used CRF classifier to detect ingredients only.

To connect the actions with corresponding ingredients we assume that cooking actions and matching ingredients occur in the same recipe instruction. We match actions and ingredients together by the following handcrafted decision model, based on the distances of the entities in the sentence and their types. We assume that all ingredients in the same instruction are mixed. If a cooking verb does not have a matching ingredient in the same instruction, we assume that this action is applied on the result of the previous instruction. Additionally, if a cooking action is an action of mixing things (e.g. "mix" and "combine") we treat the corresponding ingredients being

mixed.

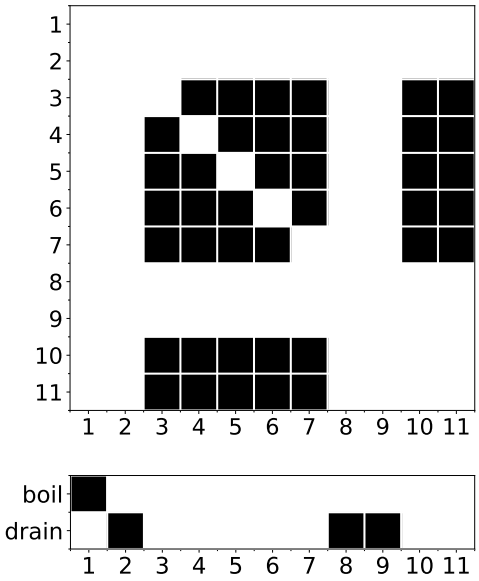


Fig. 3: Adjacency matrices for Figure 2. (mix matrix $I * I$ and action-ingredient matrix $A * I$)

As a result, a new recipe state is created that contains the used ingredients, the actions that are applied on those ingredients (up to the last processed instruction), the information which ingredients are mixed, and the state in which the ingredients are mixed. It can be seen in Figure 2 that the state is represented by appending the ingredient list and the action for each instruction. Additionally, we store the detected occurrences for each ingredient-state in an adjacency matrix (Figure 3).

After covering all the individual recipes, we merge the adjacency matrices such that we have a big set of actions, ingredients and their interrelations that can be used to build a scoring system for the unseen recipes. We create the following adjacency matrices for our scoring system.

- Matrix $A * I$: a matrix that contains the relations between actions and ingredients.
- Matrix $I * I$: a symmetrical matrix that contains the ingredients (with states) that are mixed. This helps in deciding that raw ingredients (beef) are not mixed with cooked ingredients (boiled rice).
- Matrix $A * I_{base}$: like $A * I$ but only with ingredients without their state (so instead of "noodle" and "boiled noodle" there exists a single entry for "noodle")
- Matrix $I_{base} * I_{base}$: like $I * I$ but without the states of the ingredients

For $A*I$ and $A*I_{base}$ we build another matrix that is based on cooking action groups ("heat", "prepare" and "cool") instead of specific actions so that we can assess if some ingredients are commonly "heated" or "prepared". We denote these matrices as $A_{grp} * I$ and $A_{grp} * I_{base}$. Finally, we apply a threshold on all matrix values to select the relations that occur multiple times. This information is also used in recipe evaluation later.

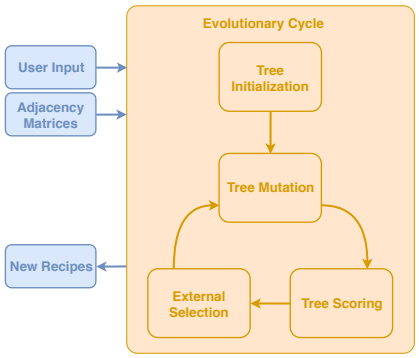


Fig. 4: Overview of the evolutionary algorithm

C. Recipe Evolution

In this section, we explain the process of creation of trees from the recipe data and evolution of recipes in search of new and optimal recipes.

1) Recipe representation: Inspired from Genetic Programming, AutoChef represents each recipe as a tree. A tree contains three types of nodes:

- Ingredient Nodes (yellow rectangle): Ingredient Nodes are the leaf nodes or terminal nodes in our tree and they store the ingredients (before any cooking action is applied to them).
- Action Nodes (blue ellipse): Action Nodes are function nodes that represent cooking actions applied on the subtree. Actions are further separated into different groups: preparation actions (e.g. "cut" or "peel"), heating actions (e.g. "cook") and cooling actions (e.g. "refrigerate")
- Mix Nodes (green diamond shape): Mix Nodes are also function nodes. They represent the action of mixing the results of their subtrees.

An example tree is shown in Figure 5.

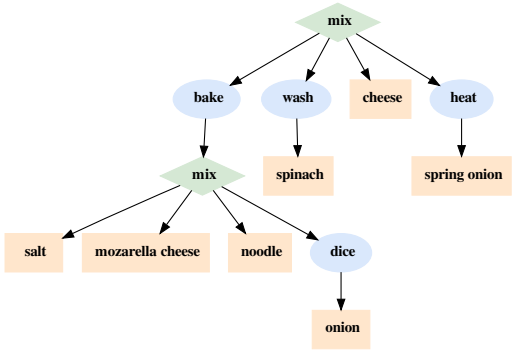


Fig. 5: Example recipe tree

2) Initial Population: To generate the initial population of recipe trees, we let the user define a list $|m|$ with at least one main ingredient and $|d|$ additional side ingredients. Given the user input n_+, we choose further ingredients that are likely to

be mixed with the ingredient lists. Using the adjacency matrix $I_{base} * I_{base}$, we select the n most mixed ingredients for each ingredient i, creating an additional list of possible ingredients. Later we merge the lists as C and keep the count of the number of occurrences $s(c)$ for each ingredient candidate $c \in C$. We build the inverse rank r for all candidates with $r(c) = 1$ for candidate $c \in C$ with the lowest sum $s(c)$ and $r(c) = |C|$ for candidate $c \in C$ with the highest sum. For each ingredient, we design the probability to be chosen as additional ingredient as

$$p(c) = \frac{r(c)}{\frac{1}{2} \cdot |C| \cdot (|C| + 1)} \tag{1}$$

Depending on the user given parameter n_+, AutoChef chooses n_+ distinct additional ingredients by the probabilities given with p from the set of given ingredients. In this process it chooses the first $\alpha \cdot n_+$ ingredients using $|m|$ and the remaining ones using $|d|$. In our experiments α is set to $\frac{1}{3}$.

We group the ingredients and choose compatible cooking actions from the set of "heating" actions. We also check whether an ingredient should be connected to a "preparing" action first. To do this, we use the action group adjacency matrix $A_{grp} * I_{base}$ to measure the likelihood of an ingredient preparation and combine it with a small gaussian distribution to add some randomness in the process. The preparation function is:

$$f_{prep}(i) = \begin{cases} 0, & if\ \frac{p(i)}{h(i)} + \chi < t \\ 1, & else \end{cases} \tag{2}$$

where i denotes the ingredient, $p(i)$ is the number of seen preparation actions on the ingredient and $h(i)$ the number of seen heating actions. $\chi \sim \mathcal{N}(0, \sigma^2)$ is a randomly chosen value with a small variance. And t is a threshold for the adjacency matrices. Value $\frac{p(i)}{h(i)}$ measures the ratio between heating and preparing actions. Similarly, we have defined a function to select a heating action as:

$$f_{heat}(i) = \begin{cases} 0, & if\ 1 - \frac{p(i)}{h(i)} + \chi < t \\ 1, & else \end{cases} \tag{3}$$

If $f_{prep}(i) = 1$ for an ingredient i we choose a valid preparation action by using the adjacency matrix $A * I_{base}$ and a rank based probability distribution as seen in equation 1. We randomly choose matching heating actions similarly. If some ingredients have the same actions, they are mixed first, then the action is applied. If the result is not a single tree yet, we merge the subtrees with a mix node as the final step.

3) Mutation: The mutation is used to update the tree structure to explore new cooking mechanics that were overlooked in the initialization phase. The mutation used by AutoChef differs by the node type as shown below. To restrict the exploration of the algorithm (e.g. we have an initial set of ingredients that must remain in the final recipe) every node can be set to be a constant. The mutation is only applied to non-constant nodes.

Mutation of Mix Nodes: For mutating the mix nodes, we separate the mix Node (n_{old}) in a "lower" (n_l) and an "upper" (n_u) mix node. Then we choose a subset of the old mix node's children (C_{old}) randomly to build the child set of the "lower" node ($C_{n_l} \subset C_{n_{\text{old}}}$). The children of the upper node are defined as the remaining set of children ($C_{n_u} = (C_{n_{\text{old}}} \setminus C_{n_l}) \cup \{n_l\}$). See Figure 6 for an example.

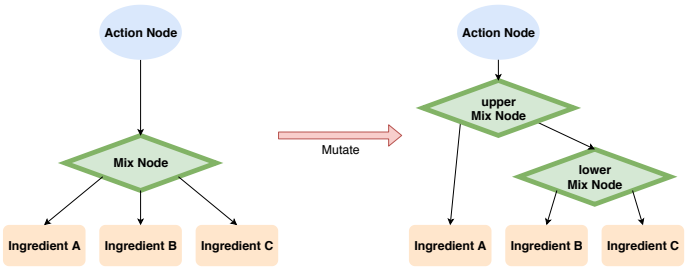


Fig. 6: Mutation of a Mix Node

Mutation of Action Nodes: For an Action Node, we can either delete the node itself (mutating the node itself) or insert a new action node at the edges as shown in Figure 7.

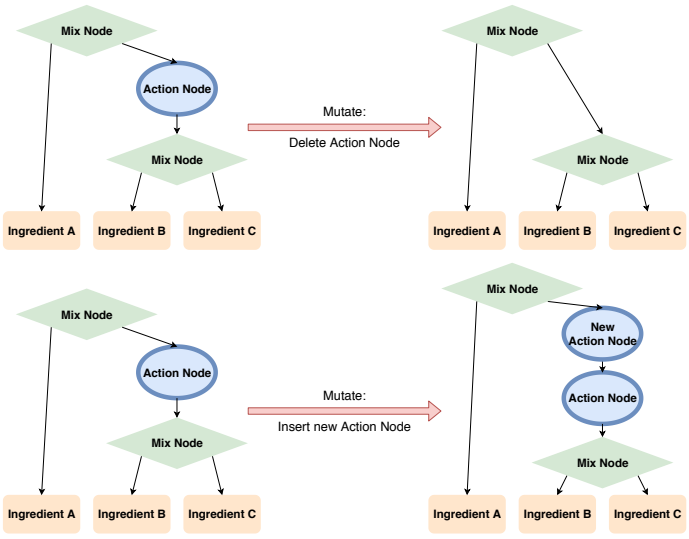


Fig. 7: Mutations of an Action Node

Note that the removal of the Action Node in the delete mutation may result in two adjacent Mix Nodes. To avoid the tree fragmentation with many mix nodes, we merge them in an additional step.

Mutation of Ingredient Nodes

For the Ingredient Nodes, we can either change the ingredient with another random ingredient or insert a random action node on the ingredient node's edge.

4) Fitness Evaluation: An ideal scoring function for a recipe is to prepare, taste and rate the meal. However, it is not practically doable to cook a whole generation and assign a fitness based on the resulting meal. AutoChef measures the recipe-tree score with the help of the initially generated adjacency matrices.

Ingredient Node Scores The ingredient nodes have three scores combined. The binary heat score $s_{heat}(i)$ for the ingredient i is defined similar to equation 3

$$s_{heat}(i, b_h) = \begin{cases} 0, & if\ 1 - \frac{p(i)}{h(i)} > t + \epsilon \wedge b_h = 0 \\ 0, & if\ 1 - \frac{p(i)}{h(i)} < t - \epsilon \wedge b_h = 1 \\ 1, & else \end{cases} \quad (4)$$

$b_h \in 0, 1$ is a binary value, indicating whether i is heated in the recipe tree or not. $2 \times \epsilon$ defines a corridor around the threshold t in which we do not punish unheated or heated ingredients. Similar to s_{heat} we can define s_{prep}

$$s_{prep}(i, b_p) = \begin{cases} 0, & if\ \frac{p(i)}{h(i)} > t + \epsilon \wedge b_p = 0 \\ 0, & if\ \frac{p(i)}{h(i)} < t - \epsilon \wedge b_p = 1 \\ 1, & else \end{cases} \quad (5)$$

we also have scores to punish duplicate actions. The first score punishes duplicate actions in the tree

$$s_a(i) = \frac{n_a(i) - n_{da}(i)}{n_a(i)} \quad (6)$$

where $n_a(i)$ defines the number of actions applied on i and $n_{da}(i)$ defines the number of duplicate actions applied on i

The final ingredient node score is then defined by the normalized sum of the scores without s_a, weighted with s_a (since this is a harder punishment than using it in the normalized sum)

$$s_{ing}(i) = \frac{1}{2} \cdot (s_{prep}(i) + s_{heat}(i)) \cdot s_a(i) \quad (7)$$

Action Node Scores For the action node score $s_{act}(a)$ we first traverse the ingredients in the subtrees of the action node. For each ingredient i_k we apply all actions upon to the node corresponding to a on it. Then we take a look in the adjacency matrix $A * I$ whether a is a valid action on the ingredient. If yes, we set $s_{i_k} = 1$, otherwise $s_{i_k} = 0$. The resulting score is

$$s_{act}(a) = \frac{1}{N} \sum_{i}^{N} s_{i_k} \quad (8)$$

Mix Node Scores The mix node's score is the normalized sum of binary values corresponding to the subtrees of the particular node. We traverse the ingredients and actions for each child subtree. Later, we build all possible pairs of ingredients from the subtree ingredient sets and lookup in our adjacency matrix $I * I$. if this is a known combination. If yes, this combination is rated with 1, otherwise 0. The final score $s_{mix}(m)$ for the mix node m is then the normalized sum of all pairwise binary ratings.

Tree Score The final tree score s_{tree} is a combination of all single node scores combined with weights to rate the overall structure of the tree. The node score is:

$$s_{nodes} = \frac{\sum_{i \in I}^{I} s_{ing}(i) + \sum_{a \in A}^{A} s_{act}(a) + \sum_{m \in M}^{M} s_{mix}(m)}{|I| + |A| + |M|} \quad (9)$$

and the final tree score is defined as

$$s_{tree} = s_{nodes} \cdot s_{duplicates} \cdot b_{acts \geq 3} \cdot b_{ings \geq 3} \quad (10)$$

where $b_{acts \geq 3}$ and $b_{ings \geq 3}$ are binary values with $b_{acts \geq 3} = 0$ if the total number of action nodes is below 3 and $b_{ings \geq 3} = 0$ if the total number of ingredient nodes is below 3. Otherwise both components are set to 1. To avoid complex recipes, we also punish the occurrence of duplicate actions:

$$s_{duplicates} = \frac{n_d}{n_a} \quad (11)$$

where n_d denotes the number of distinct actions in the tree and n_a the total number of actions. In this way, AutoChef uses the knowledge extracted from the existing recipes to score the unseen recipes.

D. Selection

For the population selection, we randomly group the new and old population pairwise and measure the tree scores. The tree with the better score is selected for the next generation. The tree score is based on tree nodes as detailed in the previous section.

E. Textual Tree Representation

Finally, the resulting recipes trees are converted into corresponding text representation for readability. We traverse each node in the tree using depth-first search and insert the node-text into predefined sentence patterns. To avoid having a lot of small instructions, we merge instructions of action nodes which have only ingredients as a child with the text of the instruction of the next layer. Additionally, if the root node is a mix Node, this mix node also generates an instruction set. As an example, the textual representation of the tree in Figure 5 is shown in Table 2.

	ingredients
	spring onion
	noodle
	salt
	spinach
	mozarella cheese
	cheese
	onion
step	**instruction**
1	dice onion and mix it with salt, mozarella cheese and noodle. Then bake it.
2	wash spinach, heat spring onion and mix it with cheese and mix it together with the results of step 1.

Listing 2: Instructions for recipe tree shown in Figure 5

IV. Experiment and Evaluation

We have created a population of 75 individuals that are evolved for 35 generations. Given the flexible usage in the ingredients and actions, we have decided to use evolutionary strategies [17]. We have used the mutation rate of $2 \cdot \frac{1}{|E|+|N|}$ where $|E|$ is the number of edges and $|N|$ the number of nodes for an individual.

We have tested our approach over several test runs and initial ingredient lists.

For the user evaluation, we selected ten random recipes with "rice" as the only input ingredient and ten with "noodle" as the only input ingredient. We created a survey in which each user was shown a few randomly chosen recipes for evaluation. The questions each user had to answer are:

1) *Is this a valid recipe?*
2) *Does it seem eatable?*
3) *Are the instructions understandable?*
4) *Is this a good combination of ingredients?*
5) *Are the used cooking actions suitable for the ingredients?*
6) *How tasty is it probably?*
7) *Would you cook it?*
8) *How creative is this recipe?*

Question 1) and 2) are binary questions, the other ones had to be answered in a range from 0 (negative) to 3 (positive). In Table 3 we present some examples from the final recipe survey pool. Recipe (a) is an instance from the set of rice recipes, recipe (b) from the noodle recipe pool. Both recipes are understandable and can be cooked to yield edible meal. Using the above questions we defined two evaluation criteria;

1) Correctness, that can be shown by combining results from questions 1,2 and 5, related to valid ingredients, edible recipe, and applied actions, and
2) Innovation, that can be reported by combining results from questions 4, 6, 8 and 7 about ingredient-combination, taste, creativity and cook-ability

Below, we present the average results over a total of 17 survey evaluations with four recipes rated in each survey. For 1) we can see in Figure 8 (a) that only 19.8 % of recipes are found to be incorrect. On the other hand, for 2), Figure 8 (b), only 6.8 % recipes are rated as non-innovative (0). Figure 11 shows the convergence of fitness over the generations. Table III(a) shows that approximately 88 - 89 % of recipes are rated as valid and edible. Table III(b) shows the average ratings over all the recipes. The table shows that AutoChef recipes have received overall good ratings with only 10% receiving lowest scores on cooking actions, and 17% for preference to cook. Similar behaviour can be observed from Figure 9(a,b). Although the recipes (both rice and noodles) are understandable, there is relatively less interest in cooking them. We attribute this trend to personal preferences and that fact that our fitness criteria favour the 'normal trend' we have converged to a more common set of ingredients and cooking methods.

	ingredients
	rice
	clove garlic
	egg
	chicken broth
	safron
	parsley
	sausage
step	**instruction**
1	beat and cut egg
2	slice clove garlic and mix it with sausage, saffron, chicken broth, parsley and rice. Then boil it.
3	Mix together the results of step 1 and step 2.

(a) a rice recipe from the survey pool

	ingredients
	zucchini
	red pepper
	water
	tomato sauce
	onion
	noodle
step	**instruction**
1	chop onion, chop zucchini and mix it with water, red pepper, tomato sauce and noodle. Then heat it.

(b) a noodle recipe from the survey pool

Listing 3: Example recipes from our survey

Recipe property	**Yes**	**No**
Validity (1)	88.42%	11.58%
Edibility (2)	89.42%	10.58%

(a) Evaluation of the binary questions

Recipe property	**0**	**1**	**2**	**3**
Understandable (3)	1.47%	5.88%	26.47%	66.18%
Tasty (6)	2.94%	27.94%	44.12%	25.00%
Would you cook it (7)	17.65%	30.88%	29.41%	22.06%
Creativity (8)	1.47%	29.41%	50.00%	19.12%
Ingredient combination (4)	4.41%	19.12%	35.29%	41.18%
Cooking actions (5)	10.29%	26.47%	41.18%	22.06%

(b) Evaluation of the non binary answers (with question numbers). The range goes from 0 (negative) to 3 (positive)

TABLE I: Average results over all recipes

V. Discussion of results and future work

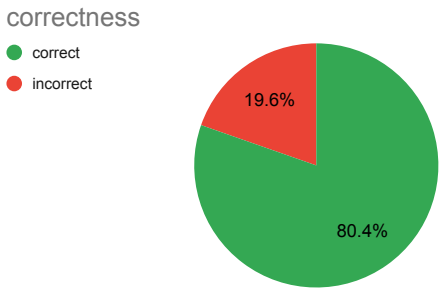
AutoChef demonstrates that it is possible to create valid recipes using an evolutionary algorithm supported by the knowledge extracted from user-generated recipes freely available on the web. Especially creating clear and understandable instructions and finding a good mix of ingredients seems to work well (Figure 8, 9 and Table I). However, it is still challenging to create recipes that fulfil personal preferences generically. A user may not be interested to cook even if the result is probably a tasty and creative meal. For future work, more data related to food replacements, spice combinations, and chemical properties of foods for better combination can be exploited to introduce interesting food combinations and cooking methods.

References

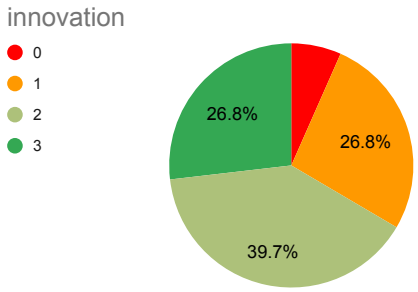
[1] "IBM Chef Watson (closed)," http://www.ibmchefwatson.com.

[2] "Autochef source code," (last visited: 2020-05-15). [Online]. Available: https://github.com/SmartDataAnalytics/AutoChef/

[3] J.-J. Chen, C.-W. Ngo, F.-L. Feng, and T.-S. Chua, "Deep Understanding of Cooking Procedure for Cross-modal Recipe Retrieval," in *2018 ACM Multimedia Conference on Multimedia Conference - MM '18*. Seoul, Republic of Korea: ACM Press, 2018, pp. 1020–1028. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3240508.3240627

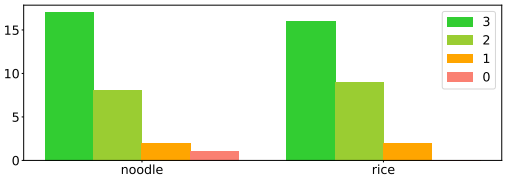


(a) The average of correctness parameters for all recipes: validation (Question 1), edibility (Question 2) and cooking actions (Question 5, all values ≥ 2)

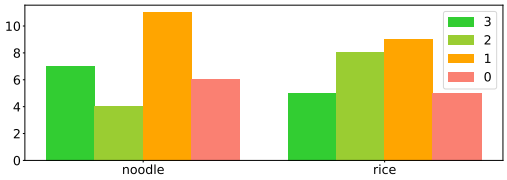


(b) Innovation measure: the average values of the answers from Questions 4,6,7 and 8

Fig. 8: Measuring correctness and innovation of the recipes



(a) Question 3: Are the instructions understandable?



(b) Question 7: Would you cook it?

Fig. 9: Evaluation results from survey. Question 3, with the highest positive ratings, Question 7, with the subjective answers

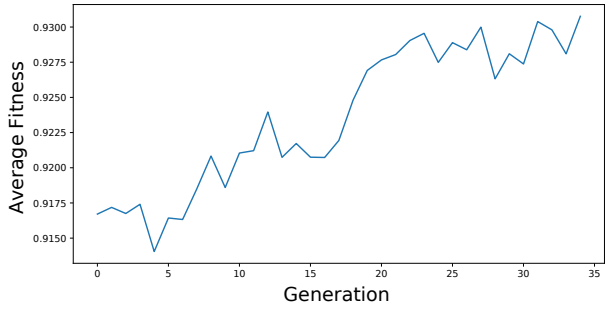


Fig. 11: Average population-fitness for 35 generations over 10 runs

[4] J. Marin, A. Biswas, F. Ofli, N. Hynes, A. Salvador, Y. Aytar, I. Weber, and A. Torralba, "Recipe1m+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images," *arXiv:1810.06553 [cs]*, Oct. 2018, arXiv: 1810.06553. [Online]. Available: http://arxiv.org/abs/1810.06553

[5] J. Freyne and S. Berkovsky, "Intelligent food planning: personalized recipe recommendation," in *Proceedings of the 15th international conference on Intelligent user interfaces, IUI '10*. United States: Association for Computing Machinery (ACM), 2010, pp. 321–324.

[6] M. Kazama, M. Sugimoto, C. Hosokawa, K. Matsushima, L. R. Varshney, and Y. Ishikawa, "A Neural Network System for Transformation of Regional Cuisine Style," *Frontiers in ICT*, vol. 5, Jul. 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fict.2018.00014/full

[7] "Bon Appetit," (last visited: 2020-01-26). [Online]. Available: https://www.bonappetit.com/

[8] "A New Kind of Food Science: How IBM Is Using Big Data to Invent Creative Recipes," Nov. 2013, (last visited: 2020-05-09). [Online]. Available: https://www.wired.com/2013/11/a-new-kind-of-food-science/

[9] "Cover:Cheese," (last visited: 2020-01-26). [Online]. Available: https://covercheese.appspot.com/

[10] E. Cromwell, J. Galeota-Sprung, and R. Ramanujan, "Computational Creativity in the Culinary Arts," p. 5.

[11] H. Jabeen, N. Tahara, and J. Lehmann, "EvoChef: Show Me What to Cook! Artificial Evolution of Culinary Arts," in *Computational Intelligence in Music, Sound, Art and Design*, A. Ekárt, A. Liapis, and M. L. Castro Pena, Eds. Cham: Springer International Publishing, 2019, vol. 11453, pp. 156–172. [Online]. Available: http://link.springer.com/10.1007/978-3-030-16667-0_11

[12] C. Draschner, J. Lehmann, and H. Jabeen, "Smart chef: Evolving recipes," EVO*, Leipzig, 2019.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

[14] "Yummly Dataset," Jan. 2017, (last visited: 2020-01-26). [Online]. Available: https://www.kaggle.com/kaggle/recipe-ingredients-dataset

[15] J. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," p. 10.

[16] "Nltk website," (last visited: 2020-01-26). [Online]. Available: https://www.nltk.org/

[17] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies–a comprehensive introduction," *Natural computing*, vol. 1, no. 1, pp. 3–52, 2002.