RECIPE GENERATOR BASED ON INGREDIENTS

Sumit Rana, Rose Kansal
Mentored by Prof. Shampa Chakraverty
Department of Computer Science
Netaji Subhas University of Technology, Dwarka Sector-3

Abstract- Cooking is the craft of preparing food with creativity and precision. It involves combining ingredients to create flavourful, nourishing meals, making it both an art and a science. In this paper, we present a personalised recipe generator tailored to individual preferences. By analysing user data like dietary needs and culinary skills, the system crafts custom recipes. We tried in making the model effective in meeting individual tastes, enhance culinary experiences, and promote healthier cooking. This recipe generator marks a significant stride in improving cooking experiences, cultural promotion and connecting users with similar culinary tastes.

1. INTRODUCTION

Food plays a crucial role in our lives, serving as a necessity, pleasure, and cultural expression. Recipe preparation merges art and science, harmonizing flavors, textures, and aromas to create delightful and nutritious meals. Different cultures boast diverse traditional recipes, reflecting unique culinary methods, ingredients, and spices rooted in geographical preferences and cultural traditions.

Translating this intricate culinary knowledge into a machine-readable format is challenging. Utilizing this data to create new recipes that blend ingredients and spices from different traditions is even more complex. Despite artificial creativity prevailing in music, images, and graphics, culinary arts have seen limited technological advancement, mainly in recipe recommendations or cuisine identification. Besides the now-defunct IBM ChefWatson [1], there are no fully autonomous recipe generators, likely due to the complexity and subjective nature of culinary outcomes.

This paper introduces the preliminary outcomes of our recipe generator, a platform integrating machine learning, natural language processing, evolutionary algorithms, and genetic programming to evolve novel recipes. Our Model analyzes ingredient actions, combinations, and generates recipes automatically. Recipes are encoded as genetic programs, evolved using specialized operators to optimize predefined fitness criteria. Figure 1 illustrates an overview of our Model (Section III), covering related works (Section II), evaluation (Section IV), and conclusions (Section V).

2. PRIOR WORK

Various research efforts focus on food preparation, cooking techniques, and flavor enhancement. For instance, investigations involve identifying recipes from images [3] [4], creating systems for suggesting recipes [5], and altering cuisine styles [6]. This article discusses automated recipe generation within computational arts. In 2014, IBM introduced "IBM Chef-Watson," generating innovative recipes by adding new ingredients to existing ones from bon-appetit [7]. Chef-Watson assessed recipe quality based on uniqueness, measured by deviation from common recipes, and aroma, by analyzing the chemical properties of flavor molecules used [8]. However, the project seems discontinued, and the

web reference is defunct. The Evolutionary Meal Management Algorithm (EMMA) [9] uses a machine learning algorithm but often lacks explicit instructions and ingredient quantities. Erol et al. [10] discovered new ingredient combinations for salads but overlooked instructions and quantities, crucial for any recipe. EvoChef [11] evolves recipes but faces limitations due to restricted data input and manual fitness evaluation. SmartChef [12] achieved semi-automated evolution, crafting more intricate recipes with partially automated fitness evaluation.

Kazama et al. [6] developed an approach transforming recipes into different culinary styles by creating embeddings for ingredients. They used word2Vec [13] to measure differences between cuisine styles, transforming recipes based solely on ingredient lists. Conversely, Marin et al. [4] considered instructions and ingredients, constructing embeddings for both. Their method involved training a neural network to learn unified recipe and image embeddings. These approaches highlight ongoing significant research in this domain. However, we are still far from a fully automated and functional recipe generator [2]

3. METHODOLOGY

Our Model 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.

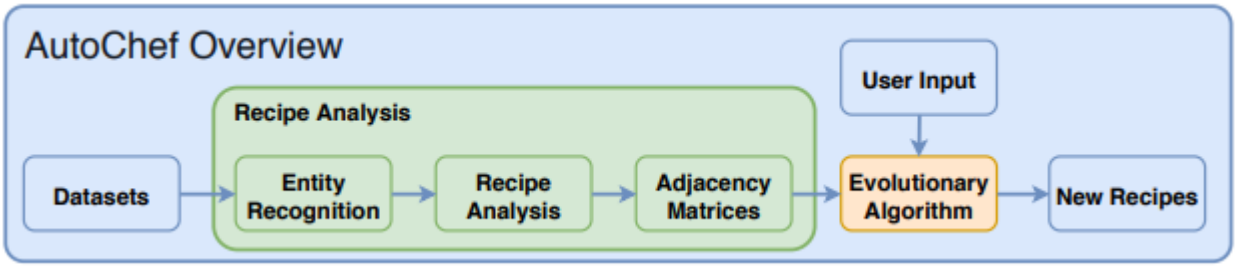


Fig. 1: Overview

3.1 DATASETS

For analysis and extracting insights regarding ingredient usage and cooking actions, two datasets have been chosen:
One Million Recipe Dataset: This collection comprises one million English recipes sourced from various popular cooking websites. Each entry includes a title, a list of ingredients, instructions, and an image of the final dish, provided as a plain text JSON array (listing 1).
Yummly Dataset: This dataset from the Yummly cooking website contains the ingredient lists of 39,774 recipes annotated with regional cuisine styles, offered as JSON.

3.2 SYSTEM ARCHITECTURE

3.2.1 RECIPE ANALYSIS

The objective is to learn the generation of novel recipes by extracting information from existing ones. To detect ingredients and instructions, a Conditional Random Field (CRF) classifier [15] is

utilized. Part-of-speech tags generated using the "nltk toolkit" [16] are employed, considering features such as sentence position, word endings, and neighboring words. These features serve as input for CRF training. Notably, ingredient recognition outperforms action recognition. Consequently, a curated list of approximately 50 fixed cooking actions is employed, utilizing the CRF classifier solely for ingredient detection.

To link actions with corresponding ingredients, a model is adopted. It assumes that both cooking actions and associated ingredients appear in the same recipe instruction. Actions and ingredients are paired through a manually devised decision model based on entity distances within sentences and their types. In cases where a cooking verb lacks a matching ingredient in the same instruction, it's assumed that the action pertains to the result of the preceding instruction. Moreover, if a cooking action involves mixing ingredients (e.g., "mix" or "combine"), the associated ingredients are treated as such.


```
{
  "ingredients": [
    {
      "text": "8 tomatoes, quartered"
    },
    {
      "text": "..."
    }
  ],
  "url": "http://www.foodnetwork.com/\
          recipes/gazpachol.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]

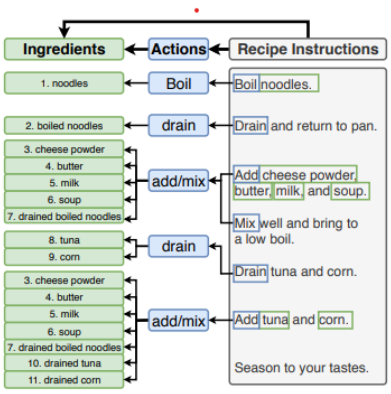


Fig. 2: Information extraction from a recipe

As a result, a new recipe state is created, encompassing utilized ingredients, actions applied to those ingredients (up to the last processed instruction), information on mixed ingredients, and the state in which they are mixed. This state representation, visible in Figure 2, appends the ingredient list and the corresponding action for each instruction. Furthermore, detected occurrences for each ingredient-state are stored in an adjacency matrix (Figure 3).

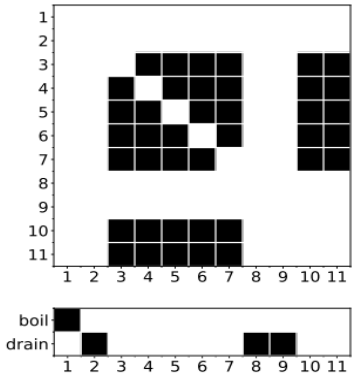


Fig. 3: Adjacency matrices for Figure 2. (mix matrix $I * I$ and action-ingredient matrix $A * I$)

Upon covering individual recipes, adjacency matrices are merged to establish a comprehensive set of actions, ingredients, and their interrelations. These matrices form the basis for constructing a scoring system for unseen recipes. The following adjacency matrices are developed for the scoring system:

- Matrix A∗I: Actions and ingredients relations.
- Matrix I ∗ I: Symmetrical representation of mixed ingredients (with states), aiding in discerning raw from cooked ingredients.
- Matrix A ∗ Ibase: Similar to A ∗ I but solely with ingredients without their states, consolidating entries like "noodle" and "boiled noodle" into a single entry for "noodle."
- Matrix Ibase ∗ Ibase: Similar to I ∗ I but without ingredient states.

For A∗I and A∗Ibase, another matrix is constructed based on cooking action groups ("heat," "prepare," and "cool") instead of specific actions, facilitating the assessment of commonly associated ingredients with these groups. These matrices, denoted as Agrp ∗ I and Agrp ∗ Ibase, undergo threshold application on all matrix values to identify recurring relations, which subsequently inform recipe evaluation.

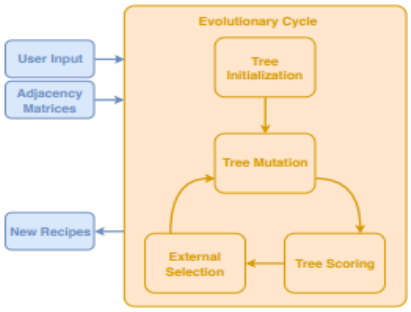


Fig. 4: Overview of the evolutionary algorithm

3.2.2. RECIPE EVOLUTION

This section outlines the process of creating trees from recipe data and evolving recipes in pursuit of novel and optimal cooking recipes.

3.2.2.1. RECIPE REPRESENTATION

- Inspired by Genetic Programming, Our Model represents each recipe as a tree, containing three node types:
- Ingredient Nodes (yellow rectangle): Leaf nodes storing ingredients before any cooking action.
- Action Nodes (blue ellipse): Function nodes representing various cooking actions categorized into preparation, heating, and cooling actions.
- Mix Nodes (green diamond shape): Function nodes depicting the action of mixing results from their subtrees.

An example tree is shown in Figure 5.

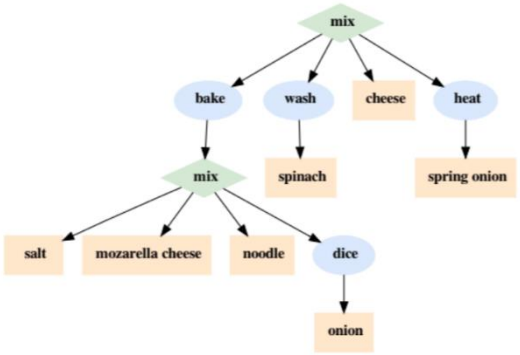


Fig. 5: Example recipe tree

3.2.2.2. INITIAL POPULATION

To produce the underlying populace of recipe trees, we let the client characterize a rundown |m| with something like one fundamental fixing and |d| extra side fixings. Given the client input n+, we pick further fixings that are probably going to be blended in with the fixing records. Utilizing the nearness framework Ibase ∗ Ibase, we select the n most blended elements for each fixing I, making an extra rundown of potential fixings. Later we blend the rundowns as C and keep the count of the number of events s(c) for every fixing up-and-comer c ∈ C. We assemble the backwards rank r for all up-and-comers with r(c) = 1 for applicant c ∈ C with the most reduced total s(c) and r(c) = |C| for up-and-comer c ∈ C with the most elevated aggregate. For every fixing, we plan the likelihood to be picked as extra fixing as

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

Contingent upon the client given boundary n+, Our Model picks n+ particular extra fixings by the probabilities given with p from the arrangement of given fixings. In this cycle it picks the first α·n+ fixings utilizing |m| and the excess ones utilizing |d|. In our trials α is set to 1/3.

We bunch the fixings and pick viable cooking activities from the arrangement of "warming" activities. We likewise check whether a fixing ought to be associated with a "planning" activity

first. To do this, we utilize the activity bunch contiguousness lattice Agrp ∗ Ibase to quantify the probability of a fixing planning and join it with a little gaussian dissemination to add some arbitrariness simultaneously. The preparation function is:

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

where I signifies the ingredient, p(i) is the quantity of seen readiness activities on the fixing and h(i) the quantity of seen warming activities. χ ∼ N (0, σ2) is a haphazardly picked esteem with a little change. What's more, t is an edge for the nearness frameworks. Esteem p(i) h(i) measures the proportion between warming and getting ready activities. Likewise, we have characterized a capability to choose a heating function as:

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

If fprep(i) = 1 for a fixing I we pick a substantial readiness activity by utilizing the contiguousness framework A ∗ Ibase furthermore, a position based likelihood dispersion as found in condition 1. We arbitrarily pick matching warming activities also. If a few fixings have similar activities, they are blended first, then the activity is applied. In the event that the outcome is certainly not a solitary tree yet, we blend the subtrees with a blend node as the last step. 3) Change: The change is utilized to refresh the tree structure to investigate new cooking mechanics that were disregarded in the introduction stage. The change utilized via Our Model varies by the node type as displayed underneath. To confine the investigation of the calculation (for example we have an underlying arrangement of fixings that should stay in the last recipe) each node can be set to be a consistent. The transformation is simply applied to non-consistent nodes. Change of Blend Nodes: For transforming the blend nodes, we separate the blend Node (nold) in a "lower" (nl) and an "upper" (nu) blend node. Then we pick a subset of the old blend node's youngsters (Cold) arbitrarily to construct the children set of the "lower" node (Cnl ⊂ Cnold). The children of the upper node are characterized as the remaining sett of children (Cnu = (Cnold \ Cnl) ∪ {nl}).

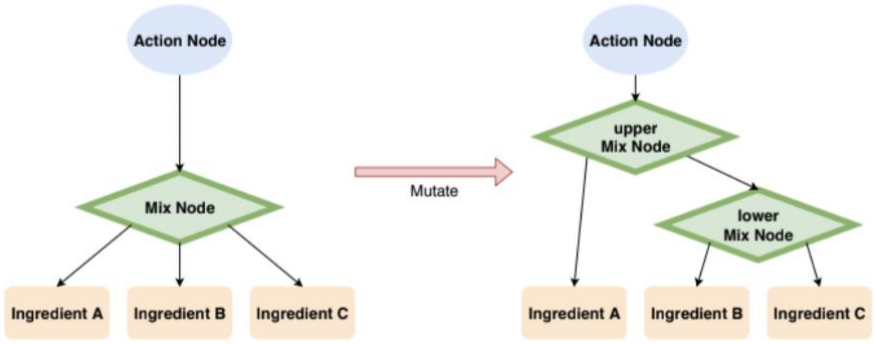


Fig. 6: Mutation of a Mix Node

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.

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. Our Model 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 sheat(i) for the ingredient i is defined similar to equation 3

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

bh ∈ 0, 1 is a binary value, indicating whether i is heated in the recipe tree or not. 2 × defines a corridor around the threshold t in which we do not punish unheated or heated ingredients. Similar to sheat we can define sprep

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

we additionally have scores to rebuff copy activities. The first score rebuffs copy activities in the tree

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

where na(i) characterizes the quantity of activities applied on I and nda(i) characterizes the quantity of copy activities applied on I The last fixing node score is then characterized by the standardized amount of the scores without sa, weighted with sa (since this is a harder discipline than involving it in the standardized sum)

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

Activity Node Scores, For the activity node score sact(a) we first cross the fixings in the subtrees of the activity node. For every fixing ik we apply all activities upon to the node relating to an on it. Then we investigate the contiguousness network A ∗ I whether a will be a substantial activity on the fixing. If indeed, we set sik = 1, in any case sik = 0. The subsequent score is

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

Blend Node Scores, The blend node's score is the standardized total of parallel qualities comparing to the subtrees of the specific node. We navigate the fixings and activities for every children subtree. Afterward, we construct all potential sets of fixings from the subtree fixing sets and query in our contiguousness framework I ∗ I. in the event that this is a known blend. If indeed, this blend is appraised with 1, generally 0. The last score smix(m) for the blend node m is then the standardized amount of all pairwise twofold appraisals. Tree Score The last tree score stree is a blend of all single node scores joined with loads to rate the by and large construction of the tree. The node score is:

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

and the last tree score is characterized as

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

where bacts ≥ 3 and bings ≥ 3 are double qualities with bacts≥3 = 0 assuming the all out number of activity nodes is under 3 and bings≥3 = 0 in the event that the absolute number of fixing nodes is under 3. In any case the two parts are set to 1. To keep away from complex recipes, we additionally rebuff the event of copy activities:

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

where nd means the quantity of unmistakable activities in the tree also, na the complete number of activities. Along these lines, Our Model utilizes the information removed from the current recipes to score the concealed recipes.

3.2.3. SELECTION

In the process of population selection, we pair up the new and existing populations randomly and evaluate their tree scores. The tree demonstrating a superior score is chosen for advancement to the subsequent generation. This score is determined based on the composition of tree nodes as elucidated in the previous section.

3.2.4. TEXTUAL TREE REPRESENTATION

Subsequently, the resultant recipe trees are transformed into text representation for ease of understanding. We navigate through each node within the tree using a depth-first search method and place the node's text within predefined sentence structures. To prevent an

excessive number of brief instructions, we merge instructions from action nodes that solely contain ingredients as children with the subsequent layer's instruction text. Furthermore, if the root node represents a mix, this node generates a set of instructions as well. For instance, the textual rendition of the tree in Figure 5 is detailed 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

4. EXPERIMENT AND EVALUATION

To evaluate our approach, we conducted multiple test runs and initiated with varied ingredient lists. For user assessment, we picked ten random recipes featuring "rice" as the sole input ingredient and another ten featuring "noodle" alone. They were prompted with specific questions:

1) Is this a valid recipe?

2) Does it seem edible?

3) Are the instructions clear?

4) Does it offer a good combination of ingredients?

5) Do the cooking actions match the ingredients?

6) How likely is it to be tasty?

7) Would you prepare this dish?

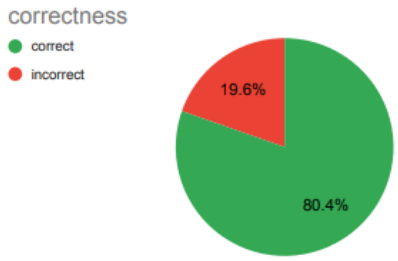
8) How innovative is this recipe?

Questions 1) and 2) required binary responses, while the others ranged from 0 (negative) to 3 (positive).

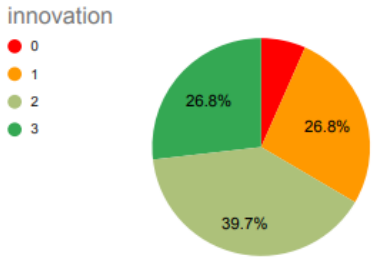
Two evaluation criteria were established using the aforementioned questions:

1) Correctness, assessed through questions 1, 2, and 5, focusing on valid ingredients, an edible recipe, and suitable actions.

2) Innovation, gauged through questions 4, 6, 8, and 7, regarding ingredient combinations, taste, creativity, and feasibility to cook.



(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

(based on our understanding of the recipes)

5. COMPARISON WITH PRIOR WORKS

Contrasting these approaches, recent research, including our ongoing efforts, aims to bridge the gaps prevalent in earlier methodologies. We integrate innovative strategies, advanced technology, and a comprehensive understanding of culinary science to elevate recipe generation. Unlike prior endeavours, our approach emphasizes a meticulous balance between creativity and practicality, leveraging sophisticated algorithms to automate the generation of intricate recipes while ensuring precise instructions and ingredient quantities.

Moreover, in contrast to earlier studies that focused solely on ingredient-based transformations [6], our methodology incorporates a holistic view by integrating both ingredient and instruction embeddings. Drawing insights from recent research [4], we are training neural networks to learn unified recipe and image embeddings, aiming for a more comprehensive and effective recipe generation system.

Despite the notable strides made in prior research, the field is yet to achieve a fully automated and functional recipe generator. Our ongoing work aims to combine the strengths of past methodologies while addressing their limitations, propelling us closer to the realization of a comprehensive, automated recipe generation system.

6. FUTURE SCOPE

Future research avenues involve incorporating additional data on food substitutes, spice pairings, and ingredient chemical properties to enhance the creation of diverse and appealing culinary

combinations and methodologies. While the model represents a significant step in automating recipe generation, the future lies in further refining this technology by incorporating a more extensive range of data sources and enhancing personalization to create diverse, culturally rich, and universally appealing culinary experiences.

1. Enhanced Data Integration: Incorporate extensive data on food substitutes, spice pairings, and the chemical properties of ingredients to broaden the diversity of recipe combinations and improve their appeal.

2. Personalization Techniques: Develop methods to personalize recipe suggestions based on individual preferences, possibly utilizing machine learning to adapt and refine recipes according to user tastes.

3. Cultural Fusion: Explore the fusion of culinary traditions from different cultures by leveraging machine-readable data on various traditional recipes, aiming to create innovative, culturally diverse dishes.

4. Objective Taste Evaluation: Investigate methods for objectively evaluating the taste and appeal of generated recipes, possibly through sensory analysis or crowd-sourced feedback mechanisms.

5. Human-AI Collaboration: Facilitate collaborations between culinary experts and AI systems to leverage human creativity and expertise while utilizing AI's computational capabilities for recipe generation and refinement.

7. CONCLUSION

The presented model demonstrates promising capabilities in generating viable recipes using an evolutionary algorithm, drawing insights from publicly available user-generated recipes. Notably successful in crafting coherent instructions and identifying effective ingredient blends, this innovation grapples with the challenge of meeting individual preferences universally.

REFERENCES

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

[2] “Autochef source code,” (last visited: 2020-05-15). [Online]. Available: https://gitnode.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

[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. Ekart, 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-distributedrepresentations-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.