A comparison between evolutionary algorithms and large language models generative approaches in the context of cooking recipes creation

Riccardo Raffini

Contributing authors: riccardo.raffini@studenti.unimi.it;

Abstract

Cooking recipes are the result of a very complex process whose final outcome is lead by creativity, comprehension and other capabilities that has long been attempted to replicate artificially in many ways, successfully. In recent years, the focus has been on promising large language models (LLMs), but there are other valid methods. In this paper, a method based on evolutionary algorithms has been explored and compared with an LMM, to see which one achieves more realistic results.

Keywords: large language models, evolutionary algorithms, generative models, cooking recipes

1 Introduction

A cooking recipe is a very specific kind of semi-structured text that describes all the necessary information to create a dish. This information includes at least, but is not limited to, a set of ingredients and a sequence of instruction steps that explain how the bare ingredients should be combined in order to obtain such a dish. However, there is no real standard in how a cooking recipe instructions are written and organized, cooking is a creative process really subjective, so this leaves us with an object difficult to handle, but once its knowledge is extracted and becomes machine understandable, there is no limit in their usage, so there is an extremely vast literature working around them, some of which aim to replicate the process of recipe creation through artificial intelligence-based techniques.

In recent years there has been a growing interest in the use of generative approaches such as **large language models** (LLMs), which became fairly simple to use but still tend to struggle with the semantic and comprehension of the creation process they try to replicate and sometimes lack the ability to take into account the user preferences because of their statistical base. An alternative generative approach can be found in genetic programming methods, more generally **evolutionary algorithms** (EA) which have been proven to be valid for different tasks in the culinary context and, more important, have the ability to include user preferences and personalization aspects in their creation process.

The aim of this paper is not to focus on one of the approaches and find a way to improve its capabilities, although some improvements in the design of an EA method were made, but rather to explore two well-affirmed techniques belonging to those approaches and compare their generative capabilities in the context of cooking recipe creation. Generally, EA are considered more traditional approaches, while LLMs are seen as modern approaches, but the two are not necessarily opposed methods, it is possible to compare

their results and at the same time spot eventual differences in their generative behaviors.

The paper is organized as follows. This first section ends with an overview of relevant generative work and other literature related to cooking recipes. A formalization of the problem and a more in-depth overview of the two selected approaches are given in section 2. The experiments performed, their setup and details are discussed in section 3. And finally, section 4 summarizes the observed experimental results and provides some ideas for improvements in the comparison and future works.

1.1 Related works

There are many research papers exploring different ideas and proposing various solutions for many cooking tasks and more in general cooking recipe processing tasks, ranging from simple classification tasks [1, 2] to more complex detection of ingredients or instructions from images or vice versa generation of images from instructions [3–5], passing through recommendation systems and recipe enhancers [6, 7]. Common to all of them is the problem of the embedding of recipes, another fundamental aspect of recipe processing; although the representation strongly depends on the solving task, many efforts have been made to find generic recipe representations that are capable of capturing the relations that occur between the ingredients [8–10].

Even if in this paper we have chosen a specific task as frame of reference, the number of possible methods and solutions does not narrow down; in fact, the broad use of artificial neural network architectures such as GPT2 [11], BERT [12], or the more general LSTM [13] and other mechanisms based on attention [14] allowed noticeably advance in this direction, in terms of both text generation and multimedia (images, videos, audio, ...). There are numerous publications in which large language models are used in the task of generating recipe text [15–17], facilitated also by the increasing number of cooking datasets available online.

But even before the diffusion of LLMs, this same task was studied and solved in more traditional ways, although some of them are also classified as artificial intelligence methods or involve their use. Evolutionary approaches have been successfully used as automatic programming tools in a wide range of applications and they are particularly useful in contexts where it is difficult to find and also model exact solutions for a problem. When narrowing the context, two significant contributions in the selected context are AutoChef [18] and EvoRecipes [19], but there are many other contributions based on different machine learning methods.

2 Recipes creation

The creation of cooking recipes is objectively a complex activity, that is because cooking is both an art and a science that in part follows some strict rules, but that is mainly guided by user creativity, a mental process that is almost impossible to perfectly replicate in an artificial way. It is, however, possible to try to replicate some aspect of it, by observing what happens during the cooking process, or as in this frame of reference, what is written in a recipe that summarizes such process and finding out which decisions and actions are commonly taken when someone creates a recipe.

It is then interesting to observe and compare how two distinct approaches, EA and LLMs behave, asking if the crescent hype in the use of LLMs is justified also for the recipe creation task, whether their use outperforms significantly the algorithmic one and if so, how good the newly generated recipes are in terms of realism, quality and style when compared to existing recipes. In this optic a generated recipe could be considered of higher quality the closer it is to its real counterpart, where the distance is measured in terms of similarity. From an abstract point of view, a perfect approach should be able to produce the same human-made recipe when prompted to use the same set of base ingredients.

2.1 Evolutionary algorithms and large language models

Even though the common idea is to learn how to create new recipes by analyzing existing recipes, it is important to state that evolutionary algorithms and large language models are based on different technologies, they present important differences that influence not only the way in which they work, but also the way in which they learn the knowledge required for the recipe creation process. The following sections show such differences

and give an overview of the training processes employed.

2.2 Evolutionary algorithms

The expression evolutionary algorithms refers to a family of similar techniques but that mainly differ by the way in which individuals in population are represented and other small implementation details that influence the flows of work; what will be described in this section is commonly referred to as **genetic programming** [20], a variant of evolutionary algorithms in which objects are usually represented by non-linear structures, e.g. tree structures, which, as it will be described, are the most natural choice for representing our subjects of interest, the cooking recipes.

Inspired by the work of AutoChef [18] and EvoRecipes [19], it is possible to design a generic two-step recipe creation process and implement it so that it can be tested against other solutions. The first step is an analysis procedure at the end of which the process becomes able to understand and detect plausible relations occurring between ingredients, actions, tools; from an abstract point of view, it is like if the method learns how to perform basic cooking activities, however, it is still not capable to combine individual activities into steps and create a full recipe. The second step is where the evolutionary approach comes in help, allowing to exploit the previously learned knowledge and fulfill the creation request, generating a new recipe from scratch and then refine it using genetic programming methods.

2.2.1 Recipe analysis

The main concept behind the first analysis step of this process is to extract information from an existing set of recipes in order to learn how recipes are commonly created. This information consists of how ingredients are combined and manipulated at different stages of the cooking process. As ingredients are transformed through the steps of a recipe, it is not sufficient to learn which actions are applied to which ingredients but also when to apply them. The *state* of an ingredient is then the set of actions that were already applied to it and that strongly influences the set of actions that may be applied next. For example, seasoning pasta is a valid action, but it is more likely that the seasoning occurs after that the pasta has been drained from boiling water; this is a common situation that simply explains why it is necessary to differentiate ingredients also by their state.

In the original AutoChef solution, actions and ingredients (and tools) were detected and extracted from textual recipes using a solution based on Conditional Random Field (CRF) classifiers [21], but their experiments have shown that this method was much better at detecting ingredients than cooking actions. Thus, in order to achieve better results in the analysis step, this component was replaced by a fine-tuned BERT token classifier [12] whose task is to perform Name Entity Recognition (NER) on the recipes instructions. The advantage of using such a component is that with just a few hundred examples manually annotated, it was possible to achieve higher precision.

This component becomes then part of a higher level *recipe parser* method that is an handcrafted procedure responsible of splitting the recipe instruction steps into shorter sentences, calling the NER component to identify entities of interest and analyze the natural language features of sentence texts, such as part of the speech and token dependencies, to create an initial simple listing representation made of tuples of the form $(action, ingredients, tools)$, where *action* is the main cooking action occurring in the considered sentence while *ingredients* and *tools* are the sets of ingredients and tools that appear to be linked to the main action. This leaves us with an intermediate listing representation of recipes that, however, is still neither able to capture the relations occurring between ingredients/tools and actions belonging to different instruction steps nor displaying the state of ingredients.

A suitable representation for recipes could be **directed acyclic graphs** (DAGs), as described in [8], and due to the sequential nature of the instructions, the graphs tend to assume a tree shape, where the leaf nodes are ingredients and tools, while the intermediate nodes are actions. Therefore the listing representation is turned in a graph connecting the entities appearing at different steps, if for some reason a tuple has no related entities or no matching is found, it is simply appended to the previous tuple. The final output of the recipe parser is then a *recipe tree*, highlighting connections between actions and ingredients but also between actions, like the one shown in

Ingredients:
Ham Cantaloupe
Instructions:
- Cut ham into 1 wide stripes and cut the cantaloupe into cubes. - Wrap a strip of ham around each cube and secure with a toothpick. - Chill and serve.

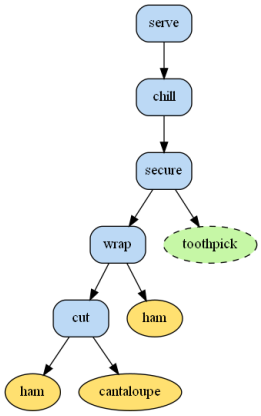

Fig. 1 Example of textual recipe and the same recipe depicted as a graph after being processed by the recipe parser.

figure 1. With this set of transformations, the state of an ingredient becomes easy to compute, as it can be derived by the set of actions encountered in the path from the ingredient node itself to the root of the tree or to the intermediate node corresponding to the currently considered instruction.

Complementary to the parsing phase, the analysis step continues with the main learning phase in which recipes tree graphs are traversed from the top root node to leaf nodes performing a deep first search (DFS), so that each ingredient state can be annotated with all the actions met in the paths and in the meanwhile detect the relations occurring between recipe entities. With the aim of learning both action-ingredient and ingredient-ingredient relations, it becomes essential to distinguish mixing actions from other kinds of action (heating, preparation, cooling, etc.). The easiest way to do this is to assign each action to its corresponding *action group* using a reference dictionary or an ontology as lookup table. All of this extracted information is stored in binary matrices, so that such knowledge becomes available in the future for both generating new recipes and evaluating them.

2.2.2 Recipe creation

Once the recipe analysis step is completed, it is possible to move on to the second step of creating recipes and evolving them using genetic programming. The generation procedure starts by asking the user for a list of main ingredients that must appear in every individual of the initial recipes population and eventually a list of additional ingredients that could be used to further expand and add more differentiation in the initial individuals. This user input and the previously compiled matrices are then used to create simple recipe trees, extracting combinations of ingredients and actions that are likely to appear in real recipes, as was learned during the recipe analysis step. Since generating recipes from simple relation matrices is a complex task and is based on random choices, the procedure may output a poor initial recipe population, where individuals may be similar between each other, and most importantly, are limited to only seen relations. Hence, there is a need for an evolution procedure that mutates and enhances those initial recipes, exploring new cooking flows and simulating the creativity of an user.

Since the evolution process is carried out through a genetic programming approach, there are no real specifications or limitations on how and in which ways the recipe trees can be evolved, all the selection criteria, mutation and/or crossover operators and fitness evaluators are strongly application dependent. However, having fixed as frame of reference the recipe creation task, one can expect to use a set of operators that aim at adding and replacing ingredients or actions, and fitness evaluators that measure the validity of actions, although there are many possibilities.

Following the usual genetic programming flow, the initial generated population of trees can be evolved applying a set of mutation and/or crossover operators, then after each evolution epoch, the original individuals together with the newly modified individuals are evaluated according to a chosen metric; in our frame of reference this possibly uses the learned relation matrices. This metric can compute individual scores for each node in the tree, according to their type (ingredient, generic action, mixing action and so on) and then combine these scores in a final one, expressing the quality of the whole graph and thus of the corresponding recipe the graph represents. The next generation individuals are then chosen according to a selection criterion, any of the commonly used criteria of evolution strategies compatible with the task objectives. Repeating this same flow epoch after epoch, one can expect to obtain a population of recipes that have higher scores than the

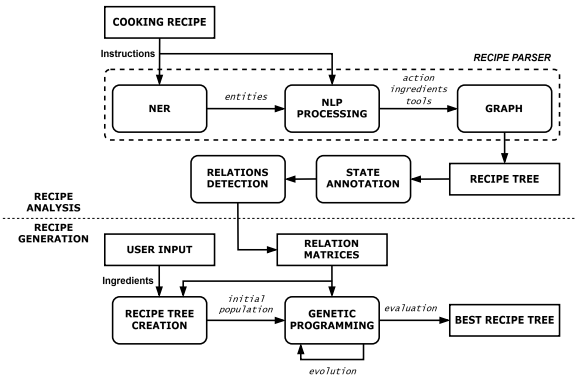

Fig. 2 Complete overview of evolutionary algorithms phases.

initial one. The exact number of epochs, the set of operators, selection criterion and other values are all hyperparameters of the evolutionary approach and will be defined in the experiments section and will determine the characteristics of the generated recipes, like the number of instruction, the prevailing kinds of action and so on.

A complete overview of the phases involved in the creation process is shown in figure 2.

2.3 Large language models

A language model is a linguistic model capable of performing natural language processing tasks, one of which is text generation. Language models are generally implemented as neural networks, with modern ones exploiting the attention mechanism [14] and transformer architectures. The adjective large refers to the fact that such models required a large number of parameters and a large amount of data for their training. Training a LLM from scratch requires, in fact, a lot of resources; thus, pre-trained models are generally employed instead, and they are then fine-tuned on the task of interest to specialize their capabilities. In our frame of reference, such task is text generation and the context of reference is cooking recipes, so by fine-tuning a model on existing recipes it will get better at creating new ones.

There are a large number of pre-trained language models to choose from, and for this comparison work, the GPT2 [11] model was chosen. It is a transformer-based pre-trained model, that regardless of its relatively small number of parameters, is able to achieve good results in sequence generation as it was specifically trained to predict sequence of words in sentences. A transformer-based model follows a specific encoder-decoder architecture and uses an attention mechanism to elaborate the input sequences, capturing the relations occurring among elements in the sequences and outputting probabilities.

In this optic a cooking recipe is just a sequence of sentences, each of which is in turn composed of a sequence of words but arranged in a certain order to form a meaningful cooking recipe. The aim of training the chosen model becomes to make it understand such an order. Compared to the evolution process, the language model does not need to understand the specific meaning of the words, nor the fact they represent ingredients or actions; of course, text generation requires a semantic understanding to compose meaningful sequences, but this is all handled by the underlying statistical model.

As stated, the training of the chosen language model is conceptually easier and faster because there is no need of introducing a recipe analysis phase as done for evolution process; as LLMs are data-driven methods, they learn directly from the sequence of words of the given examples, so in this case it is enough to provide recipe instructions to the model, with the only caution of transforming raw text tokens (individual words) into numerical sequences. That is because, as with any other machine learning method, it processes numbers rather than text. The transformation is performed using an index mapping defined over a vocabulary of words and embeddings; additionally, some special tokens, required to identify the parts composing the recipe, are inserted in the original sequence. Some of these transformations can be performed automatically through a tokenizer.

GPT2 is part of the more wide category of predictive language models that work completing sequences of tokens by observing the previously generated ones. For this kind of models, the output is represented by an individual token that represents the most probable token that could be inserted after the previously seen sequence of tokens, so to generate a full sequence, it is necessary to call the model multiple times, until the desired length of sequence is reached or a special termination token is predicted.

When the model is employed for generating a new sequence, the next token generation is driven by an initial short sequence of tokens that commonly include the user input (prompt), but its content is very problem-specific. In our creation task, a new recipe generation can be started by providing the initial list of ingredients that should appear in the recipe; the model will then be able to complete the rest of the recipe, that is, generate the instructions composing the recipe. Instead of what happens for genetic programming, the language model is also responsible for determining how long the generated recipes will be, although it is possible to specify a maximum length.

2.4 Textual representation of methods output

A shared characteristic of both the evolutionary method and the language model is that they produce raw outputs, respectively, a recipe graph and a sequence of tokens; although quite easy to interpret and to understand, they differ from the original and usual textual representation of a cooking recipe. Their readability can be significantly improved by parsing and adapting the obtained outputs into a semi-structured text, which lists the set of ingredients and the sequence of instruction steps, as one would expect a written cooking recipe to be.

The language model formatted output is easier to obtain, it is sufficient to convert token indices to the corresponding text tokens and look at the special tokens to split the generated sequence in ingredients and instructions. Converting a recipe tree generated by the evolution process is instead more complex, the form of tree makes them easier to manipulate, but only essential information regarding entity connections is stored in them. A possible way of obtaining a textual representation consists of traversing the tree following a depth first search and simply creating an instruction line for each action node, naming connected ingredients or previous steps when required.

3 Experimental results

In this section, we present an experiment performed on a selected dataset of recipes. The goal was to setup and train the two creation processes described in the previous section and then use them on a common set of inputs to generate recipes that could be used to accomplish the main comparison task.

3.1 Datasets

There exist many datasets aggregating data on cooking recipes obtained from different sources, having many features including, of course, ingredients, quantities, instructions, but also titles, descriptions, categories, preparation times, etc. However, for the end of the simple comparison proposed, not all those features are required, it is sufficient to have a dataset whose recipes have a list of ingredients and instructions, a category and user numerical ratings. Starting from the two related *Food.com* recipes and reviews datasets [22, 23], after some simple data manipulation, it was possible to obtain a dataset of approximately 200.000 recipes that meet the requirements, each with an aggregated numerical rating whose values range from 0 to 5, where 0 is the lowest level of user appreciation while 5 is the highest. However, due to the limit of computational resources available, during the experiment, training phases were carried out on a smaller subset of recipes of size 60.000 randomly extracted, whose relevant information are reported in figure 3.

To perform the comparison task, a small set of 100 recipe examples that were not included in the training dataset were chosen as the reference dataset to evaluate the creativity of the two selected approaches. The idea is to have some sets of ingredients to use as prompts for method generation, while it is assumed that the associated instructions were created by users, that they are the result of their natural creative cooking processes, and that such user processes were also prompted only by the set of ingredients. The characteristics of this dataset are reported in figure 4; as can be seen, only recipes with at most 12 instruction steps and at most 13 ingredients were selected to simplify the generation and comparison.

3.2 Generating recipes with evolutionary algorithms

To test the effectiveness of recipe creation through the evolutionary algorithm approach, the analysis step of the creation process described in section

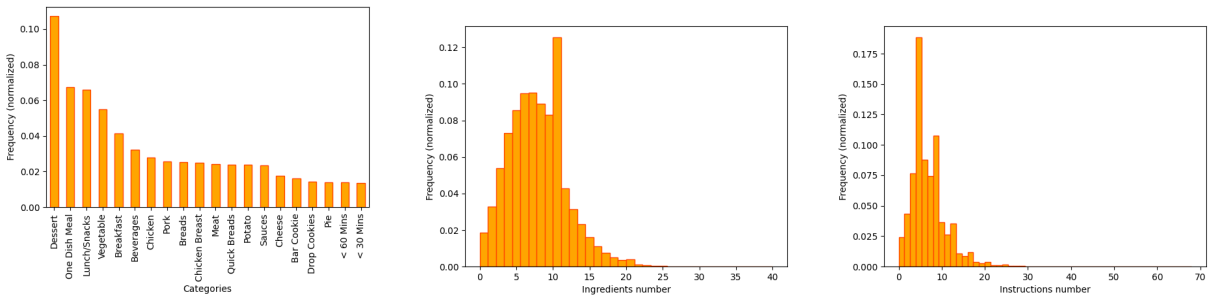

Fig. 3 Characteristics of the training dataset. From left to right: top 20 categories, distribution of ingredients number and distribution of instructions number.

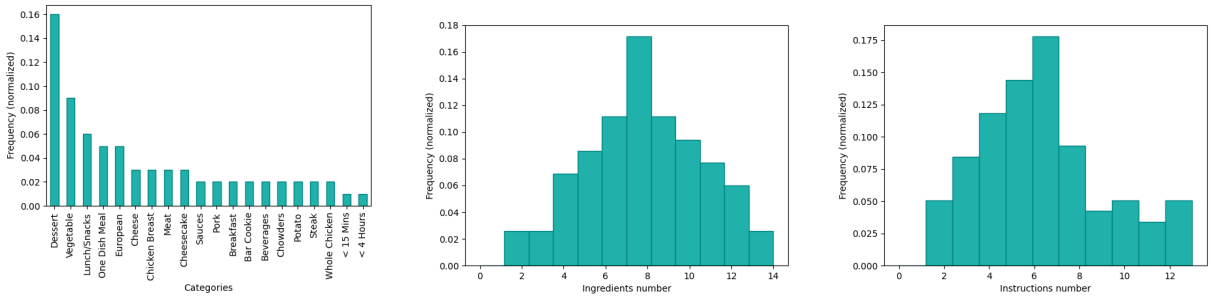

Fig. 4 Characteristics of the evaluation dataset. From left to right: top 20 categories, distribution of ingredients number and distribution of instructions number.

2.1, was run on the train dataset. The NER model employed in the recipe parser is a pre-trained version of BERT [12] downloaded from the Hugging Face platform, trained for the token classification task using 800 cooking recipes randomly extracted from the Recipes3K dataset [24] and manually annotated with *food*, *tools* and *action* entity labels. The result of this analysis phase is a collection of ingredient-action and ingredient-ingredient relation matrices that can be used in the next phase for the evolution and evaluation of recipes after their generation.

For each reference recipe, from the evaluation dataset, the genetic programming algorithm was set up to generate an initial population of 10 recipe trees, generated limiting only to the main ingredients provided in the input. The evolution flow was then run for 10 epochs, using the same fitness evaluator described in Autochef [18], an elitism criterion for population selection, and a set of mutation operators that can insert, replace, remove action nodes, add ingredient nodes, or add tool nodes. The best individual of the final population is returned as the output of the creation process and converted into its corresponding textual representation. Running the generation process for each ingredient set in the reference dataset, leaves us with 100 algorithms-generated recipes.

3.3 Generating recipes with language models

At the same time, to test the effectiveness of creating recipes using a large language model approach, a pre-trained version of the GPT2 model downloaded from the Hugging Face platform [25] was trained on the same train dataset.

Before starting with model training, a *tokenized dataset* of examples was created as follows. The original 60.000 recipe examples were split into train/test partitions with a 95/5 ratio. Each raw recipe text was transformed into a sequence of textual tokens, together with special control tokens and the expected user prompt; this sequence composes the recipe structure that model should learn:

the input and the ingredients segments both listing the names of the ingredients occurring in the recipe, and the instructions segment, listing the instructions for creating the recipe. The complete dataset was then obtained passing textual tokenized examples through the *tokenizer* associated with the GPT2 model, returning the corresponding numerical token sequences, and finally concatenating these sequences to build blocks of tokens of size 1.024 (this last step helps speed up the training process).

The model was trained using this tokenized dataset for 20 epochs on a NVIDIA GeForce RTX 4060 Ti cuda-capable GPU, using a train batch size of 4 and *Adam* optimizer with a learning rate equal to 5e−5.

After training, the model was asked to create new recipes from the sets of ingredients in the reference dataset, but they first needed to be turned into valid tokenized input segments to guide the recipe generation. The model was also limited to a maximum sequence length of 1.000. This second generation process also leaves us with another 100 model-generated recipes.

3.4 Evaluation

In a similar way, the creative processes of evolutionary algorithms and large language models were prompted to create recipes both on the same lists of ingredients, but each of them followed its own creation process. By adding the layer described in 2.4 at the end of both creation processes, it was possible to get a uniform text structure for recipes and thus share a common evaluation criterion. At the end of the generation phases, the results are 100 pairs of artificially generated recipes and 100 natural (real) recipes, one for each set of ingredients in the reference dataset.

The similarity of recipes was then evaluated in a **vector space** created from the corpus of documents represented by instructions of the real recipes in the reference evaluation dataset. In this space, every recipe can be turned into the corresponding vector by computing the **TF-IDF** score for each term appearing in its instructions. Having a vector representation for recipes allows one to easily determine the similarity between a generated recipe and a reference recipe as the inverse of their vector distance, computed by any distance function, such as **cosine similarity**. Then

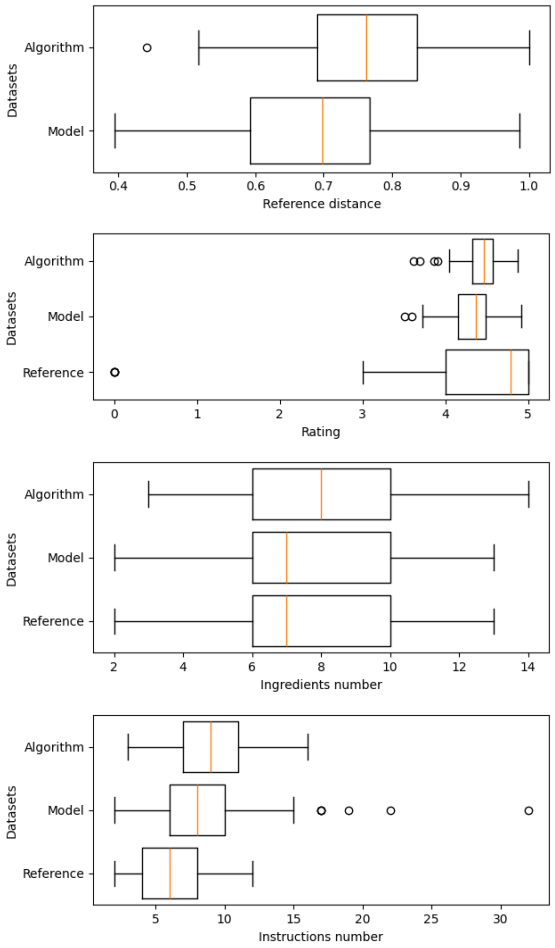

Fig. 5 From top to bottom, distributions of: reference recipes and generated recipes distances, computed rating of generated recipes and reference rating, ingredients number, instructions number.

each generated recipe was rated and categorized by aggregating the values of their 10 nearest neighbors, found using the same distance function and in the vector space. The rating was calculated as the average of neighbor ratings weighted by their distances, while the category was determined as the most common category among neighbors, if no category prevails over the other, the recipe is simply labeled as *uncategorized*. Other global observations on the characteristics of the generated recipes were also made.

Figures 5 and 6 report respectively the most significant comparison results and the category distributions in the generated recipes.

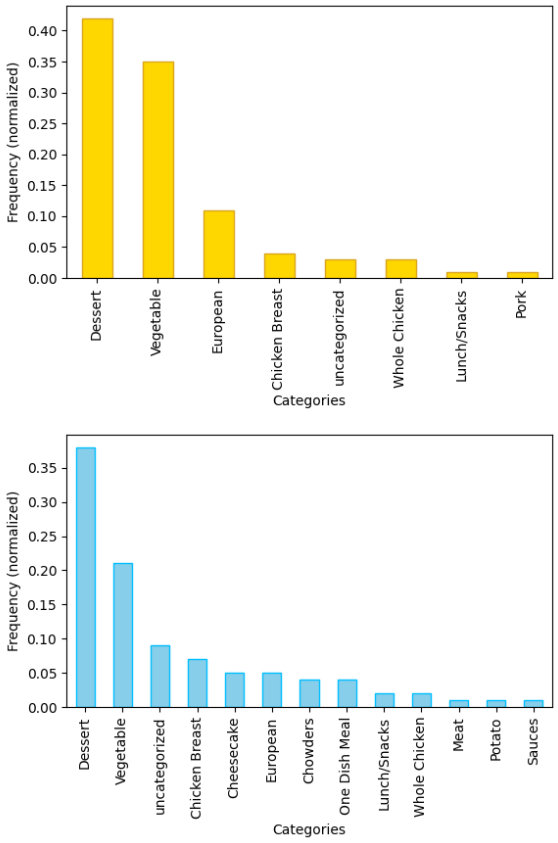

Fig. 6 Top, categories distribution of algorithm-generated recipes. Bottom, categories distribution of model-generated recipes.

4 Conclusions and future work

The results have shown that the creative process of language models generates recipes more similar to the reference recipes than the one generated by the evolutionary algorithm, as can be seen by the distance distributions reported in the top graph of figure 5. In the 80% of reference case considered, the model recipe was closer to the real recipe than the algorithm one.

However, this distance gap is not very large, and it is probably due to the text transformation method used to make the recipes uniform. In fact, turning a recipe tree into a textual recipe is complex, so poor implementation may have significantly influenced the results. Table 1 shows a comparison between a reference recipe and the pair of recipes generated by the two approaches; as can be seen, the evolutionary algorithm recipe has simpler instruction steps, with very few terms to use for vectorization.

When considering how many times the reference recipe is the closest neighbor of the generated recipes, it turned out that 64% of the algorithm-generated recipes have the reference as the closest neighbor, while only 46% model-generated recipes have it.

From an overall point of view, instead, it can be observed that the artificially generate recipes have similar characteristics as reported in figures 5 and 6. From these data, it can be seen that the categories distributions of the methods resemble the reference distributions in figure 4, but in reality, only 25% of the generated recipes share the same category with the reference recipes. It can also be observed that the methods tend to have a higher number of instructions compared to the reference recipes.

All these considerations imply that it cannot be stated with absolute certainty which method creates the best recipes and which prevails over the other. However, if we only look at the main question of which approach has the most realistic creative process, then the answer is that large language models have a better creation process than the one achieved with evolutionary algorithms because model recipes are more similar to the real counterparts, but none of them is able to achieve the same quality and style.

Despite that, it cannot be ruled out that by refining the evolutionary algorithm process proposed, for example, by carrying out cross-validation on the algorithm parameters to find the configuration values that provides the best results, or by training the language model for more epochs, the same quality levels will be achieved.

Other future work that can improve this comparison could be exploring whether the focus on certain recipe categories or the use of different datasets may have a significant impact in recipe creation or in their evaluation. This can also help to state whether the results are due to other unconsidered factors. Or even see if choosing other models and/or architectures gives similar results.

References

[1] Su, H., Lin, T.-W., Li, C.-T., Shan, M.-K., Chang, J.: Automatic recipe cuisine classification by ingredients. In: Proceedings of

Shared ingredients		
chili powder, cumin, thyme leaves, red pepper flakes, boneless chicken breasts		
Reference recipe	Model recipe	Algorithm recipe
- In a large bowl, combine the onion recipe soup mix, bread crumbs, chili powder, cumin, thyme, and pepper. - Dip the cubed chicken in bread crumb mixture, coating well. - In a large skillet, heat 1/2-inch oil over medium temperature and saute chicken, turning once, until done; drain on paper towels. - Serve warm and, if desired, with assorted mustards and other condiments (blue cheese dip or buttermilk dressing are good).	- Preheat oven to 425 degrees. - Spray a large cookie sheet with cooking spray. - Cut chicken breast into pieces. - In a small bowl, stir together onion soup mix, spices, and dry dressing. - Add chicken to bowl; toss well. - Place 1 chicken breast piece on each piece of chicken. - Roll chicken tightly. - Place, seam side down, in a greased baking dish. - Cover dish with foil and bake at 425 degrees for 25 to 30 minutes. - Turn chicken once during the cooking. - Combine topping ingredients and spoon over each serving.	- Mash the cumin; parboil it; preheat it. - Cook a red pepper flakes. - Burn boneless chicken breasts then thicken it. - Saute a thyme leaves. - Toss in them. - Beat together some chili powder and bake in it. - Mix them.

Table 1 Comparison between reference, model-generated and algorithm-generated recipes.

the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication. UbiComp '14 Adjunct, pp. 565–570. Association for Computing Machinery, ??? (2014)

[2] Sharma, T., Upadhyay, U., Bagler, G.: Classification of cuisines from sequentially structured recipes. In: 2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW) (2020)

[3] Chhikara, P., Chaurasia, D., Jiang, Y., Masur, O., Ilievski, F.: Fire: Food image to recipe generation. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pp. 8184–8194 (2024)

[4] Salvador, A., Drozdzal, M., Giro-i-Nieto, X., Romero, A.: Inverse cooking: Recipe generation from food images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

[5] Pan, S., Dai, L., Hou, X., Li, H., Sheng, B.: Chefgan: Food image generation from recipes. In: Proceedings of the 28th ACM International Conference on Multimedia. MM '20, pp. 4244–4252. Association for Computing Machinery, ??? (2020)

[6] Teng, C.-Y., Lin, Y.-R., Adamic, L.A.: Recipe recommendation using ingredient networks (2012). https://arxiv.org/abs/1111.3919

[7] Yanai, K., Maruyama, T., Kawano, Y.: A cooking recipe recommendation system with

visual recognition of food ingredients. International Journal of Interactive Mobile Technologies **8**(2) (2014)

[8] Jermsurawong, J., Habash, N.: Predicting the structure of cooking recipes. In: Conference on Empirical Methods in Natural Language Processing (2015)

[9] Li, D., Zaki, M.J.: Reciptor: An effective pretrained model for recipe representation learning. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1719–1727. Association for Computing Machinery, ??? (2020)

[10] Tian, Y., Zhang, C., Metoyer, R., Chawla, N.V.: Recipe representation learning with networks, pp. 1824–1833. Association for Computing Machinery, ??? (2021)

[11] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., *et al.*: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)

[12] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (long and Short Papers), pp. 4171–4186 (2019)

[13] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[14] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[15] H. Lee, H., Shu, K., Achananuparp, P., Prasetyo, P.K., Liu, Y., Lim, E.-P., Varshney, L.R.: Recipegpt: Generative pre-training based cooking recipe generation and evaluation system. In: Companion Proceedings of the Web Conference 2020, pp. 181–184 (2020)

[16] Bień, M., Gilski, M., Maciejewska, M., Taisner, W., Wisniewski, D., Lawrynowicz, A.: Recipenlg: A cooking recipes dataset for semi-structured text generation. In: Proceedings of the 13th International Conference on Natural Language Generation, pp. 22–28 (2020)

[17] Goel, M., Chakraborty, P., Ponnaganti, V., Khan, M., Tatipamala, S., Saini, A., Bagler, G.: Ratatouille: a tool for novel recipe generation. In: 2022 IEEE 38th International Conference on Data Engineering Workshops (ICDEW), pp. 107–110 (2022). IEEE

[18] Jabeen, H., Weinz, J., Lehmann, J.: Autochef: Automated generation of cooking recipes. In: 2020 IEEE Congress on Evolutionary Computation (CEC), pp. 1–7 (2020)

[19] Razzaq, M.S., Maqbool, F., Ilyas, M., Jabeen, H.: Evorecipes: A generative approach for evolving context-aware recipes. IEEE Access **11**, 74148–74164 (2023)

[20] Bartz-Beielstein, T., Branke, J., Mehnen, J., Mersmann, O.: Evolutionary algorithms. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **4**(3), 178–195 (2014)

[21] Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the Eighteenth International Conference on Machine Learning. ICML '01, pp. 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)

[22] Food.com Recipes with ingredients and tags. https://www.kaggle.com/datasets/realalexanderwei/food-com-recipes-with-ingredients-and-tags

[23] Food.com - Recipes and reviews. https://www.kaggle.com/datasets/irkaal/foodcom-recipes-and-reviews

[24] Recipes 3K (2023). https://www.kaggle.com/datasets/crispen5gar/recipes3k

[25] GPT-2 Hugging Face Platform. https://huggingface.co/openai-community/gpt2