ChefLution: An Evolutionary Inspired Path for Enhancing Multi-Agent Systems

Sam Brennan
sbrennan@utexas.edu
University of Texas at Austin
Austin, Texas, USA

Salvador Robles
salvadorrh@utexas.edu
University of Texas at Austin
Austin, Texas, USA

ABSTRACT

We provide a systematic approach to enhancing multi-agent systems by integrating principles from Evolutionary Computation and leveraging the capabilities of Large Language Models (LLMs). In our framework, LLM agents act as independent evolutionary operators, facilitating the creation and iterative refinement of solutions. As a case study, we introduce ChefLution, an evolution-inspired multi-agent system designed to generate novel recipes. ChefLution incorporates components such as initial population creation, crossover, mutation, and an evaluator that scores recipes on a scale from 1 to 100. To benchmark its performance, we compare ChefLution against two baseline systems: one consisting of a recipe creator and evaluator, and another incorporating an initial population creator, crossover, and evaluator. ChefLution significantly outperforms both baselines, demonstrating its ability to produce higher-quality and more creative recipes.

In addition, we investigate the role of evaluation strategies within our system. Our findings reveal that LLM-guided evaluation using few-shot prompting is superior to zero-shot and chain-of-thought prompting, offering both greater consistency and accuracy in scoring. These contributions highlight the potential of evolutionary-inspired frameworks powered by LLMs to enhance creativity and problem-solving in multi-agent systems.

ACM Reference Format:
Sam Brennan and Salvador Robles. 2024. ChefLution: An Evolutionary Inspired Path for Enhancing Multi-Agent Systems. In *Proceedings of Neural Networks course (CS 394N '24)*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Since the advent of LLMs, researchers and practitioners have dedicated countless hours to discovering how the groundbreaking technology can be harnessed and used effectively. One such method is multi-agent systems where instances of LLMs are used as "agents" that interact with one another to collaboratively solve complex tasks, refine ideas through iterative feedback, or take on specialized roles in problem-solving scenarios. Early research has showcased the potential of multi-agent LLMs to contribute to all fields of computer science, especially computational creativity. The field of computational creativity seeks to utilize artificial intelligence to solve tasks traditionally associated with human creativity, such as recipe generation and refinement. Recipe refinement, in particular, is a task that balances creativity with objective and practical constraints, such as taste, novelty, visual appeal, and feasibility (difficulty in making it). Although traditional evolutionary algorithms have shown strong performance in creative tasks, they rely on heuristic-driven mechanisms that do not fully exploit domain-specific knowledge. This project does not seek to replace or contradict these methods, but rather to explore how multi-agent systems of LLMs can perform alongside traditional approaches by incorporating domain knowledge and task specialization. LLMs, with their ability to generate and process human-like text, offer an opportunity to enhance evolutionary processes by assigning specialized roles to individual agents, creating a more nuanced and adaptable system for tasks like recipe generation and refinement.

Another method in which LLMs have been used is that of an evaluator. Evaluating creative outputs, such as recipes, often relies on human judgment, which can be subjective, time-consuming, and resource-intensive. LLMs have opened up the opportunity to automate this process by taking the place of the human evaluator. However, challenges remain in ensuring that LLM evaluations align with human preferences and avoid common pitfalls, such as biases or inconsistencies. To determine the most effective approach for LLM-guided evaluation, different prompting strategies will be implemented and analyzed, each offering unique strengths.

By leveraging these prompting strategies, it becomes possible to explore the strengths and weaknesses of LLM-guided evaluations in creative problem-solving tasks. If LLMs can be shown to serve as reliable evaluators across various domains, they could greatly streamline the evaluation process. This would reduce the need for time-consuming and costly human evaluation while enabling more consistent and objective assessments. Moreover, solving key challenges—such as aligning LLM judgments with human preferences, mitigating biases inherited from training data, and ensuring robustness across diverse contexts—could further enhance their reliability. In doing so, LLM evaluators could be seamlessly integrated into broader AI systems, such as multi-agent frameworks that combine evaluation with generation and optimization processes, ultimately enabling more efficient and scalable solutions for tasks requiring subjective judgment.

This paper investigates two main ideas that build upon each other. The first main idea explores whether LLMs can serve as reliable evaluators in creative tasks such as recipe generation. With recipe generation, an LLM evaluator would assess recipes for qualities such as taste, novelty, visual appeal, and feasibility. If LLMs prove to be reliable evaluators, the next step is to identify the most effective prompting strategy to optimize their performance in this role. The second main idea builds upon this by investigating whether a multi-agent system of LLMs, structured to simulate computational evolution, can address the challenges of creative problem solving. Specifically, we propose a system in which specialized LLM agents take on different roles in an iterative evolutionary process: generating an initial population of recipes, evaluating them against predefined criteria, and applying mutations and crossovers to refine and optimize them. By combining the strengths of multi-agent LLMs' creative capabilities with the iterative improvement inherent in evolutionary processes, this approach aims to generate recipes that score highly on taste, novelty, visual appeal, and feasibility. Together, these ideas represent a novel approach to creative problem-solving, with implications not only for recipe generation but also for broader applications in artificial intelligence and evolutionary computation.

Therefore, we pose the following research questions:

- *What is the best framework in which we can create an evaluator, that resembles a fitness function from Evolutionary Computation, to evaluate a multi-agent response?*
- *Can ChefLution (our evolutionary inspired solution) create novel and better recipes than other baseline multi-agent systems? How effective is ChefLution in comparison with different evaluation mechanisms?*

2 BACKGROUND

This section discusses four main topics that form the foundation of this project:

(1) LLMs as Evaluators: How LLMs have been used to assess the quality of outputs in various domains, including their strengths, limitations, and different prompting techniques that perform well at this task.
(2) Multi-Agent LLMs: Their success in tasks like problem-solving and world simulation, and their untapped potential as evolutionary operators.
(3) Computational Evolution: How an LLM-based evolutionary process can excel in tasks that require nuanced understanding or deep domain-specific expertise.
(4) Integrating LLM Evaluation and Multi-Agent LLMs: Synthesis of these two approaches and their relevance to this project, particularly in using LLMs for creative domains such as recipe generation.

Each topic will be reviewed in turn, with an emphasis on their contributions and shortcomings, before concluding with how they connect to the domain of recipes.

2.1 LLMs as Evaluators

The first topic discusses what's known as "LLM-guided evaluation," where LLMs asses the quality of the outputs generated by other LLMs [1,3]. An LLM is responsible for generating text in response to a prompt, and a different instance of an LLM is responsible for evaluating the response. This approach has been applied in areas like text summarization, where LLMs evaluate the coherence, fluency, and informativeness of summaries generated by other LLMs. Similarly, LLMs have also been used to evaluate in creative domains, where they assess novelty and relevance. This project exists in the creative domain. The following techniques have shown promise in LLM-guided evaluation:

- **Zero-shot learning**: the model is asked to perform a task without having been explicitly trained or shown any example for that specific task. The model uses its general knowledge to infer how to complete the task based on the prompt alone. For example, asking the LLM to rate a recipe with no prior example of a recipe being rated.
- **Few-shot learning**: the model is provided with several examples of how to complete the task, helping it understand the specific patterns or objectives for the task at hand. This allows the model to tailor its response more accurately than in zero-shot settings. For example, giving an LLM an example of several recipes and their rating, then asking it to rate a new recipe.
- **Chain-of-thought reasoning**: the model is prompted to work through the steps of reasoning out loud, allowing it to break down the problem into smaller parts. This is particularly useful for tasks that require multiple steps or logical progression. For example, an LLM prompted to use chain-of-thought reasoning when analyzing a recipe would logically step through each criteria (taste, novelty, visual appeal, and feasibility) explaining to itself how different flavor combinations and instructions influence each score.

Although these techniques have increased performance in LLM-guided evaluation, they are not without limitations. LLMs may inherit biases from their training data, produce hallucinations, or deviate from intended behavior. The biases LLMs inherit from their training data can lead to poor evaluations. For example, an LLM might give higher scores to recipes that are long and have complex ingredients, while ignoring short, simple and tasty recipes simply because these patterns were more frequent in its training data. In addition, LLMs tend to optimize for surface-level coherence rather than deep semantic understanding. This can result in evaluations that favor outputs that "sound right" but fail to critically assess their quality in more nuanced ways. For example, an LLM might give a high score to a well-structured recipe with unusual flavor pairings, like anchovies, and chocolate, because it is technically well written. Also, LLMs may struggle with consistency when given vague evaluation criteria. For example, an LLM that's told to evaluate a recipe might give it different scores depending on if it's favoring taste, texture, aroma, presentation, or any other characteristic of "good" recipes.

2.2 Multi-Agent LLMs

The second main topic of the background section discusses multi-agent LLMs. Multi-agent LLMs are an innovative approach in computer science, designed to extend the functionality of large language models by enabling them to collaborate within structured frameworks. Instead of relying on a single model to perform all tasks, multi-agent systems employ multiple instances of an LLM, each specializing in a specific role or task. These agents work together to tackle complex challenges that require diverse expertise and coordination.

Multi-agent systems have been found to perform well on a variety of complex tasks that span many different fields. Specifically, research has shown high performance in two realms: problem solving and world simulation [2]. In the problem solving realm, each agent specializes in one field, and the agents collaboratively work together to solve a complex problem. This is similar to how people with different expertise work together to solve a problem in the real world. Research has shown multi-agent LLMs perform well at problem solving tasks such as software engineering, robot modeling, and science experiments. The success of multi-agent LLMs in world simulation stems from each agents' ability to role-play. Each agent takes on the identity of an entity living in the world, outlined by the designer, and the resulting behavior is observed. Research has been specifically done with societal simulation, gaming, psychology, economy, policy making, and disease propagation simulation. This work was designed to leverage multi-agent systems ability for problem solving. Multi-agent systems are a new area in computer science and there is still much that is not known about them.

Our work is closely related to EvoRecipes [6], which also explores the generation of novel recipes using Evolutionary Algorithms. However, our approach diverges significantly by leveraging multi-agent systems powered by LLMs as evolutionary operators. To the best of our knowledge, we are the first to utilize LLMs in this capacity, introducing a unique framework for enhancing the evolutionary process.

2.3 Computational Evolution

The third main topic of the background introduces the branch of computer science called computational evolution. Computational evolution applies principles from biological evolution to solve complex optimization and creative problems. Evolutionary algorithms (EAs) are inspired by the principles of natural selection and evolution, where populations of solutions evolve over time through processes like selection, mutation, and crossover. These algorithms are widely used in optimization and creative problem-solving, particularly when the solution space is vast and poorly defined. Computational evolution enables iterative improvement by maintaining a diverse population, assessing fitness based on defined criteria, and generating new solutions by combining or modifying existing ones. While EAs have been shown to be successful, they typically rely on rigid and random mutation and crossover operations. In tasks where nuanced understanding or domain-specific expertise exists, LLMs offer an alternative approach. LLMs and multi-agent systems can enhance traditional EAs, bringing advanced reasoning, linguistic understanding, and task specialization to the evolutionary process. By integrating these approaches, ChefLution seeks to extend the capabilities of computational evolution, applying it to creative domains like recipe generation.

Research supports the idea of incorporating LLMs into the evolutionary process. In Evolution through Large Models, an instance of an LLM was used to perform "advanced mutations" when generating programs. It relied on the LLM's understanding of code structures and logical operations to produce meaningful variations [4]. This showcases how LLMs can effectively assist evolutionary processes by leveraging their internal reasoning and domain knowledge to create more targeted and intelligent modifications, suggesting a potential role for LLM-based mutations in domains like recipe generation or creative problem-solving. Connecting Large Language Models with Evolutionary Algorithms Yields Powerful Prompt Optimizers, evolutionary algorithms were applied to enhance prompts given to LLM. Meaning crossover and mutation, applied with evolutionary algorithms, were shown to increase the quality of the prompts. Multi-agent evolution seeks to achieve similar results through the use of LLMs instead of evolutionary algorithms.

2.4 Computational Evolution and Multi-agent LLMs with LLM-Guided Evaluation

Taken together, these concepts lay the groundwork for this paper, which aims to explore a novel intersection of LLM-guided evaluation, multi-agent systems, and evolutionary processes within the domain of recipe generation. LLMs' ability to evaluate outputs in creative tasks provides a scalable alternative to human judgment, while multi-agent systems enable task specialization and collaboration, mimicking real-world problem-solving dynamics. Evolutionary approaches, in turn, offer a structured mechanism for iterative improvement, aligning naturally with the creative process of refining recipes. By integrating these domains, this work investigates whether multi-agent LLMs can function effectively as evolutionary operators, with an evaluator agent playing a central role in assessing fitness. This synthesis not only advances the application of multi-agent systems in creative problem-solving but also addresses open questions about the reliability and objectivity of LLM-guided evaluators in subjective domains like recipe generation.

2.4.1 Why Recipes? Recipes are an excellent domain for this endeavor for a variety of reasons. Firstly, recipes are linguistic by nature, leveraging the strengths of LLMs in generating and interpreting natural language. In addition, recipes strike a balance between structure and creativity. Recipes are structured in the sense that it has clear criteria such as taste, novelty, visual appeal, and feasibility. This provides a concrete framework for evaluating fitness, ideal for LLM-guided evaluation. On the other hand, recipe generation offers infinite possibilities for innovation, from combining unconventional ingredients to exploring unique cooking techniques, making it a rich creative problem solving task, perfect for an evolutionary framework. By focusing on recipes, this project addresses a domain that is both fun and challenging, with results that can be evaluated through both objective and subjective measures.

3 APPROACH

This section discusses how the evaluation method will be chosen and also how multi-agent evolution will be implemented, as well as the role for each of our agents: Initial Population Agent, Fitness Agent, Mutator Agent, and Crossover Agent). We aim to provide a comprehensive overview of the multi-agent evolutionary framework and its implementation in the context of recipe optimization.

3.1 Evaluation Approach

A reliable evaluator must be consistent and accurate. In the context of recipe evaluation, consistency is the ability for the evaluator to score the same recipe in the same way and similar recipes in a similar way. Accuracy is the ability for the evaluator to score recipes similarly to how expert human judges would score them. Without an evaluation system that thrives in each area, the evolutionary process risks favoring suboptimal or irrelevant solutions, undermining the quality of the final outcomes. Ensuring the evaluator's robustness requires thorough testing against diverse inputs and careful prompt engineering. Only with a trusted evaluator in place can meaningful experiments and comparisons proceed. System messages for zero-shot prompting, few-shot prompting, and chain-of-thought reasoning were created to maximize the consistency and accuracy for each type of evaluator. Reference the appendix to see the specific prompts. A system message is merely the first message given to an agent to establish the agent's identity and task. The prompting technique that is the most consistent and accurate was chosen to be used as the system message for the fitness agent.

3.2 Multi-Agent Evolution Approach

Now, we will introduce the flow of dialogue and describes the role of each agent. The evolutionary process consists of several distinct steps, each of which will be one agent in a multi-agent system organized as shown in Figure 1.

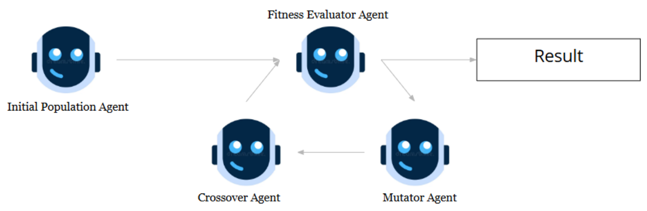

Figure 1: The flow of recipes follows this order. They are first generated by the initial population agent, then iteratively evaluated, mutated, and crossed over. Not all recipes will be both mutated and crossed over during one iteration.

Initial Population Agent. One agent will be responsible for generating an initial population of recipes. In evolutionary computation, optimized initial population generation results in a diverse population spread across the solution space. This results in more exploration and reduces the likelihood of premature convergence to suboptimal solutions. Likewise, this agent will create recipes much different from one another in terms of ingredients and cooking instructions. By introducing diversity early on, the agent increases the chances of discovering creative and unconventional recipes that may satisfy the optimization criteria. This agent's ability to generate a broad and varied initial population sets the foundation for a robust evolutionary process, enabling subsequent agents to refine and improve upon high-potential solutions more effectively.

Fitness Agent. The second topic describes how the fitness function will be implemented as LLM-guided evaluation. A key component of evolutionary computation is a good fitness function. Its sole purpose is to determine how good an individual is in the population by evaluating it against predefined objectives. A well designed fitness function not only ensures the selection of high-quality candidates but also encourages diversity and creativity within the population. This is where LLM-guided evaluation will be used. The fitness agent was given clear objectives for it to rank the recipes. This agent is prompted to score each recipe from 1 to 100 for each category: taste, novelty, and feasibility. Several LLM-guided evaluation methods will be compared to identify the most effective approach for this project. The comparison will include: zero-shot prompting, few-shot-prompting, and chain-of thought reasoning.

Mutator Agent. In evolutionary computation, small changes, or mutations, are made to highly fit individuals prior to reevaluation in hopes of improving what has already been found to be good and exploring new areas of the solution space. Therefore, a mutator agent must take recipes that perform well and make a small modification to them with the goal of improving its score. The mutator agent will be able to make advanced mutations because it leverages its internal reasoning and domain knowledge to create more targeted and intelligent modifications, ensuring that changes are both meaningful and aligned with the goals of improving taste, novelty, visual appeal and feasibility. It will also have access to the score or each recipe prior to mutation so it will know which area should be targeted for improvement. By balancing refinement with novelty, the mutator agent ensures the evolutionary process remains dynamic and continuously produces innovative solutions.

Crossover Agent. In evolutionary computation, crossover combines elements from two or more high-performing individuals to create new solutions that inherit the strengths of their parents, yet different from the current population. In recipe refinement, a crossover agent will combine two high performing recipes with the hope of improving upon the recipe. Specifically, this will take the form of combining like ingredients and instructions. For example, if one recipe contains rice chicken, lemon sauce, and instructions to grill the chicken and another has noodles, beef, and spaghetti sauce, with instructions to pan fry the beef, an offspring could be noodles, beef and lemon sauce with instructions to grill the beef. Like the mutator agent, the crossover agent will have access to previous scores and will be able to make smart crossovers with the goal to improve those scores.

3.3 Approach Conclusion

This section outlines the structure and functionality of the multi-agent LLM-based evolutionary process. Each agent—Initial Population, Fitness, Mutator, and Crossover—plays a distinct role in driving innovation and improvement, while the system message ensures their coordination. By leveraging LLM capabilities, this approach addresses challenges in recipe optimization, such as balancing creativity with feasibility. Although this framework offers a promising method for tackling nuanced tasks, its effectiveness will ultimately depend on the implementation and refinement of each agent's contributions. This sets the stage for evaluating the success of this system in subsequent experiments.

4 EXPERIMENTS

This section discusses how we address our research questions:

(1) RQ1: Evaluator strategies comparison. Explanation of specific criteria being evaluated with different prompting strategies. We will also test our hypothesis that evaluator performance will be better in certain prompting strategies.
(2) RQ2: ChefLution against other solutions. Explanation of the different evolutionary frameworks being evaluated. Then the experiments will be conducted testing the hypothesis that multi-agent evolution is an effective method of evolution.

We implemented our experiments using a cloud server machine with Intel(R) Xeon(R) CPU @ 2.20ghz. The system has 12.7 GB RAM, and the disk space is 225.8 GB, leveraging its cloud-based environment for efficient computation and scalability. The Large Language Model used in this work was GPT-4o mini, accessed via the OpenAI API.

This was done to determine the best method for LLM evaluation and determine the effectiveness of multi-agent evolution. Baselines were first established, then different methodologies were attempted. The first topic is discussed here: evaluation experimentation setup, execution, and results.

RQ1: Evaluator strategies comparison

4.1 Evaluator Testing and Results

To facilitate testing, three datasets were created, each containing 50 recipes to represent different quality levels: one set of bad-quality recipes, one of average-quality recipes, and one of excellent-quality recipes. To establish consistency, each prompting strategy was tested against the same 3 recipes (one from each category). For each trial, the evaluator was told to evaluate the same recipe 100 times. 5 Trials were completed for each recipe/prompting strategy combination and the standard deviation is recorded in Table 1 across all trials. In the eyes of consistency, it's not important what the scores were, but rather, how much the scores change.

The three-shot and chain-of-thought method performed nearly identically in consistency, achieving very low standard deviations with average and excellent recipes. Additionally, they both performed poorly with the bad recipes. This issue stems from the limitations of their training data, as there is a lack of data specifically focused on "bad" recipes. Most culinary datasets are biased toward higher-quality, standardized recipes. Models trained primarily on positive examples may struggle with understanding the nuances of what makes a recipe "bad" in a systematic way, leading to more variable or inconsistent evaluations.

In the context of multi-agent recipe evolution, the performance of an evaluator on bad recipes is less critical because such recipes are unlikely to appear frequently within the system. The idea is that bad recipes, being far removed from successful or viable ones, will be filtered out or evolve into more acceptable solutions over time. The evaluator's primary function in this context is to help guide the evolution of good recipes, so its accuracy and consistency in evaluating those recipes are far more important than its performance on bad ones. As long as the evaluator can adequately assess the key traits of average recipes, its occasional misjudgment of bad recipes will not significantly affect the overall evolutionary process. Moreover, as the evolution progresses, the population of bad recipes naturally shrinks, reducing the evaluator's exposure to these "edge cases" and allowing it to focus on evaluating higher-quality solutions.

Given that: zero-shot performed poorly with average and excellent cases, performance in the bad recipe category is less of a concern, and three-shot and chain-of-thought performed nearly identically well, three-shot and chain-of-thought prompt strategies are the most consistent with this task.

To establish accuracy, A series of trials were conducted with each prompting strategy. Each trial consisted of scoring 50 recipes from a specific category (Bad, Good, or Excellent) and comparing their classification against the correct criteria. Incorrect classification occurred when a recipe's total score fell outside its designated range. Recipes were classified as "Bad" if their score was less than 34, "Good" if their score was between 34 and 66 (inclusive), and "Excellent" if their score was between 67 and 100 (inclusive). Five trials were conducted for each prompting strategy and each recipe category. Table 2. reports the mean and standard deviation of misclassifications across the five trials.

The three-shot prompting strategy is clearly the most accurate, having the least amount of misses in each category of recipes.

Additionally, all prompting strategies performed the worst in the average recipe category. This is because average recipes, lacking extreme qualities, require a more nuanced evaluation and are thus more susceptible to variability in scoring. There is also a more fine line between an "average" recipe and an "excellent" one. When all models missed, they tended to be missing high. This is evident by very high accuracy in the excellent category by all models. This could be why models that performed worse on average recipes' accuracy test but well in the consistency test.

As can be seen, the chain-of-thought evaluator performed by far the worst in every category except for excellent recipes. Its errors were the result of over scoring the recipes. Despite being prompted to be "harsh and brutally honest" it inflated most average recipes to excellent status. It was noticed that the visual appeal rating was skewed upwards when compared to other recipes. This could have been the result of how the model interpreted the given definition of visual appeal. The results in the table reflect several iterations of prompt engineering where improvement was made, but it still performs poorly.

Answer RQ1: The three-shot prompting method, on the other hand, exhibited the fewest misclassifications across all categories. This success can be attributed to the specific examples it was given, which helped the model better understand the scoring criteria and context. By providing a few relevant examples, the model was able to grasp patterns of evaluation more effectively and consistently apply them to new recipes. Because this model is the most consistent and accurate, it will be used as the evaluator prompt in the multi-agent evolution framework.

	Bad Recipe	Average Recipe	Excellent Recipe
Zero-Shot	4.79	4.40	7.81
Three-Shot	6.91	2.84	2.73
CoT	6.81	2.85	2.43

Table 1: This table illustrates the consistency of each method. One recipe from each category was evaluated 100 times with each method and the resulting standard deviations of each scoring is documented in the table.

RQ2: Performance of ChefLution compared to alternative Multi-Agent Solutions

To evaluate the performance of ChefLution against alternative multi-agent systems in generating novel recipes, we conducted experiments where each system was tasked with creating recipes using a fixed set of ingredients: Bread, Cheese, Pepperoni, and

	Bad Recipe		Good Recipe		Excellent Recipe	
	Mean	SD	Mean	SD	Mean	SD
Zero-Shot	2.00	2.82	7.80	3.70	1.00	1.73
Three-Shot	0.20	0.44	6.8	1.92	0	0
CoT	6.40	5.17	10.80	1.92	0	0

Table 2: This table illustrates the accuracy of three methods—zero-shot, three-shot, and chain-of-thought—in evaluating recipes across three categories: Bad, Good, and Excellent. The values in the table represent the mean and standard deviation of misclassifications across all trials.

Tomato. By standardizing the ingredients across experiments, we ensured a consistent basis for comparison, minimizing variability that could arise from differing input sets and preventing overly divergent recipe outputs.

We tested each system's performance with two different evaluator configurations. Based on insights from RQ1, the few-shot prompting strategy was selected as the primary evaluator due to its demonstrated effectiveness in providing accurate and consistent scoring. Additionally, we included a zero-shot prompting evaluator to explore a more resource-efficient alternative. This approach considers practical constraints in multi-agent systems, such as token limitations, where zero-shot strategies may offer advantages in certain scenarios. Both evaluator evaluate a recipe on a scale of 1 to 100. By including both evaluation methods, we aimed to provide a comprehensive assessment of ChefLution's capabilities relative to the baseline systems.

4.2 Chef-Judge Baseline (Two-Agent)

We are going to consider two baselines, one that is very simple and that is a bit more complex with more different agents as evolutionary operators. The first baseline is a simple two-agent system designed to provide a reference point for comparing more complex configurations. In this baseline, one agent serves as a "chef" responsible for generating recipes, while the other acts as a "judge," evaluating the recipes based on the predefined criteria: taste, novelty, visual appeal and feasibility. This straightforward setup represents the most basic form of multi-agent iterative improvement, where the chef and judge interact in a simple loop where the chef generates one recipe, then the judge evaluates, and the chef refines in an iterative fashion. This model allows for the comparison of more advanced systems involving multiple specialized agents, providing a controlled environment to measure the impact of incorporating domain-specific expertise and task specialization through additional agents. The simplicity of this two-agent system provides a clear starting point, ensuring that any observed improvements in more complex systems can be attributed to the added complexity rather than the basic structure itself.

Chef-Judge Baseline Results (Two-Agent) For the first evaluator (zero-shot), as shown in Figure 2, the initial recipe score begins at 50 and improves to 60 by the second generation. However, the system stagnates thereafter, with no further improvements observed in subsequent generations. Similarly, for the second evaluator (few-shot), as illustrated in Figure 3, the initial score is approximately 35, increasing to 50 after the second generation, but remaining constant across the following iterations.

These results demonstrate that the two-agent system lacks the ability to improve solutions over time, becoming stagnant after the second iteration with no further gains in recipe scores. This limitation highlights the need for more sophisticated frameworks to achieve sustained performance improvements across generations.

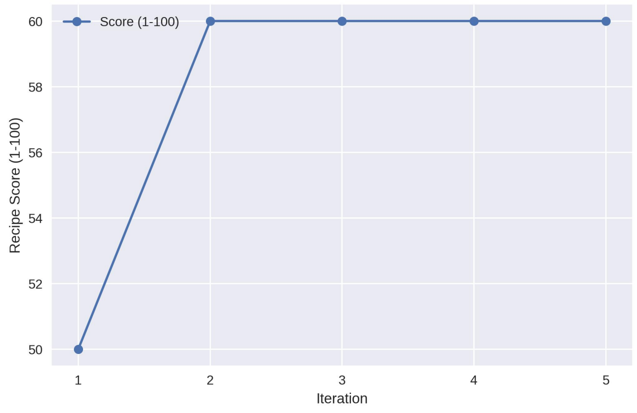

Figure 2: Performance of chef-judge two-agent system, using the zero-shot evaluator. Baseline showing not a lot of improvement across iterations.

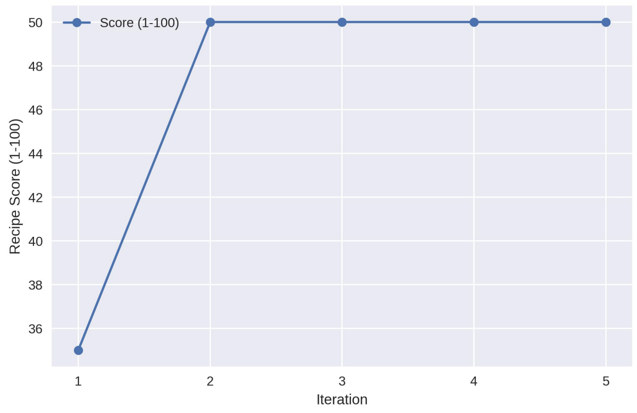

Figure 3: Performance of chef-judge two-agent system, using the few-shot evaluator. Baseline showing not a lot of improvement across iterations.

4.3 Crossover Baseline (Multi-Agent System)

The second baseline consists of a more sophisticated multi-agent system incorporating additional agents compared to the first baseline. This system begins with an initial population agent, which generates a diverse pool of k recipes. For this experiment, $k = 4$, resulting in an initial population of four recipes.

Next, a crossover agent creates new recipes by combining existing ones. This agent generates all possible pairwise combinations of the current population, performing crossover to produce $\binom{k}{2}$ new recipes. In this case, with $k = 4$, the crossover agent generates six recipes.

Once the new recipes are created, an evaluator agent scores each recipe on a scale of 1 to 100. For all experiments, the evaluator operates in either a zero-shot or few-shot prompting mode. After evaluation, the recipes are ranked based on their scores, and the top k recipes (equal to the size of the initial population) are retained to form the new generation. This iterative process of crossover, evaluation and selection is repeated for five iterations to get our final pool of recipes.

This baseline more closely mimics evolutionary computation and their core principles. The process being with a population of candidate solutions (recipes), applies variation through crossover to generate new candidates, evaluates the fitness of each candidate (via scoring), and selects the fittest individual to carry forward to the next generation.

Crossover Results (Multi-Agent System) Figures 4 and 5 illustrate the performance of the two-agent multi-agent system utilizing a crossover approach to generate recipes. As previously described, six recipes are generated in each iteration, and their scores are plotted in the figures. In each iteration, Rank 1 corresponds to the highest-scoring recipe, while Rank 6 represents the lowest-scoring recipe.

With the zero-shot evaluator (Figure 4), the initial population of recipes exhibits a diverse range of scores, spanning from 45 to 75—an improvement of 15 points over the first baseline's initial population. However, by the final iteration, the system stagnates, with all six recipes scoring 75. While no further improvements are observed beyond this point, the initial diversity of solutions clearly contributed to the system's better starting performance compared to the first baseline.

In contrast, the results using the few-shot evaluator (Figure 5) show a less diverse initial population, with scores ranging from 45 to 60. Despite this lower starting range, the crossover approach demonstrates its potential over successive iterations, gradually improving the population's scores. By the final iteration, recipe scores increase to a range of 65 to 70—representing a 20-point improvement over the first baseline.

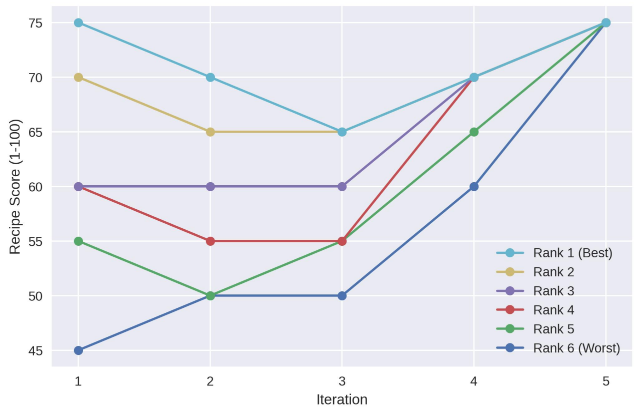

Figure 4: Performance of crossover multi-agent system, using the zero-shot evaluator. Population of recipes achieving all 75 scores by the end of the 5th iteration.

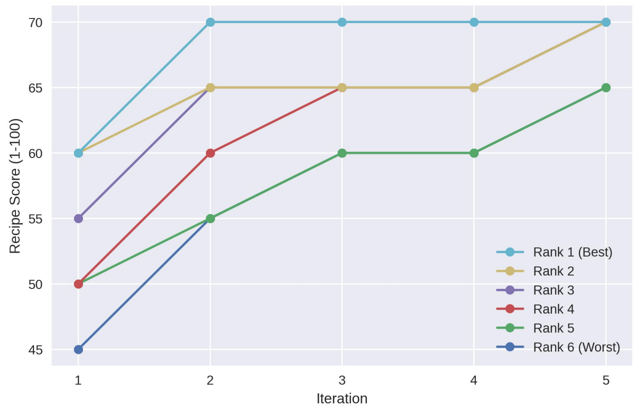

Figure 5: Performance of crossover multi-agent system, using the few-shot evaluator. We show a more consistent growth of recipe scores across the population.

4.4 ChefLution: Complete Evolutionary Cycle

To achieve a fully realized evolutionary algorithm, ChefLution introduces a critical component missing from the second baseline: a Mutator agent. This agent operates by making slight modifications to existing recipes, aiming to enhance them by introducing incremental improvements.

ChefLution is an evolution-inspired multi-agent system that begins with an initial population agent generating a diverse pool of recipes. Each recipe in this initial population undergoes mutation by the Mutator agent, introducing variations to create potentially better recipes. The Crossover agent then performs pairwise combinations of recipes to generate new ones by recombining their features. Finally, all recipes are evaluated using the Evaluator agent, which assigns a fitness score on a scale of 1 to 100. The top-performing recipes are selected to form the next generation, and the process of mutation, crossover, evaluation, and selection is repeated iteratively.

This framework expands upon the second baseline by incorporating mutation as an additional evolutionary operator, making the system more robust and capable of exploring a wider solution space. By combining mutation, crossover, and selection, ChefLution effectively mimics the complete evolutionary process, driving the generation of higher-quality and more creative recipes. The results of this enhanced approach are presented in Figure 6 and Figure 7.

ChefLution Results (Multi-Agent System) Similar to the crossover baseline, ChefLution generates six recipes per iteration, reflecting the number of recipes produced by applying crossover to an initial population of four recipes.

When using the zero-shot evaluator (Figure 6), the initial population exhibits scores ranging from 50 to 70, comparable to the crossover baseline. However, unlike the crossover baseline, ChefLution demonstrates steady and consistent improvement in recipe scores across iterations. By the final iteration, the recipe scores range from 90 to 95—an increase of 20 points compared to the crossover baseline. This indicates that the recipes generated by ChefLution perform exceptionally well, showcasing the system's ability to refine and enhance solutions over successive iterations.

With the few-shot evaluator, the initial population starts with scores ranging from 45 to 60. As the process progresses, ChefLution steadily improves the quality of the recipes, culminating in a final set of scores ranging from 80 to 85—an improvement of 15 points over the crossover multi-agent baseline. This demonstrates the effectiveness of incorporating a complete evolutionary algorithm, including agents for mutation, in achieving superior results.

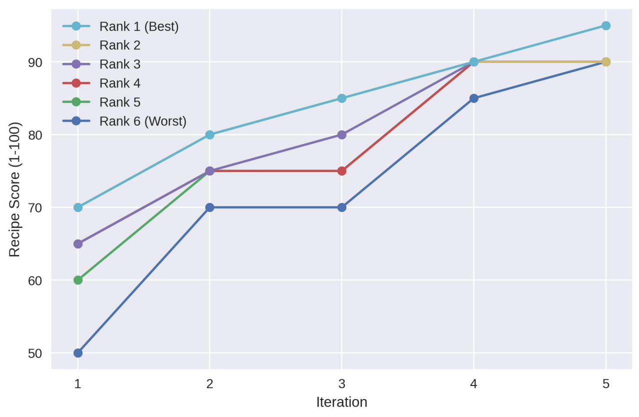

Figure 6: Performance of ChefLution, using the zero-shot evaluator. We demonstrate a set of high-performing set of recipes.

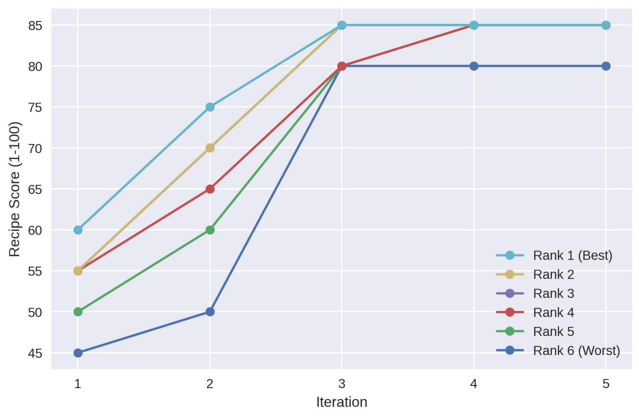

Figure 7: Performance of ChefLution, using the few-shot evaluator. We demonstrate a diverse set of recipes that keep improving.

By integrating all key components of traditional evolutionary algorithms—mutation, crossover, evaluation, and selection—ChefLution systematically refines recipes with each iteration, leading to consistent gains in recipe quality. Compared to the second baseline, which includes only crossover and evaluation, ChefLution's inclusion of the Mutator agent enables it to explore the solution space more effectively and avoid getting trapped in local minima. This allows ChefLution to generate higher-quality recipes that are both novel and well-optimized according to the evaluator's scoring.

A side by side comparison of the two baselines and ChefLution can be seen in Figure 8 and Figure 9. These results highlight the effectiveness of a "complete" evolutionary-inspired algorithm, which consistently outperforms less sophisticated systems in the task of creating novel recipes. Additionally, our findings demonstrate that ChefLution continues to improve over successive iterations, making it a robust framework capable of avoiding suboptimal solutions and refining its outputs. This robustness emphasizes ChefLution's ability to maintain steady progress and deliver high-quality recipes, even in challenging optimization scenarios.

Answer RQ2: Our results clearly show that ChefLution outperforms both baselines. As anticipated, the addition of a sophisticated, fully inspired evolutionary algorithm leads to significant improvements in the fitness scores of generated recipes. Specifically, ChefLution surpasses the second baseline (crossover multi-agent system) by the second iteration. By the final generation/iteration, ChefLution achieves a performance increase of over 15 points compared to the Crossover baseline when using the few-shot evaluator, and an improvement of over 20 points when using the zero-shot evaluator.

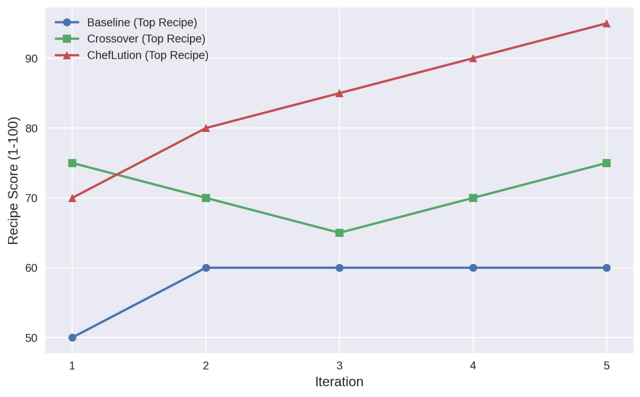

Figure 8: ChefLution vs Baselines using zero-shot evaluator.

5 DISCUSSION AND FUTURE WORK

LLM-based evaluation with few-shot prompting has proven to be a consistent and accurate method of evaluation. Future work can still be done to further optimize this kind of evaluation. Chain-of-thought reasoning proved its consistency but did not perform nearly as well as few-shot in accuracy. Many iterations were completed when prompt engineering for chain-of-thought and the evaluators kept improving. Further prompt refinement may make this method more competitive with the few-shot method. The details of chain-of-thought make it more customizable, but also more difficult to optimize. For example, with recipe evaluation, the chain-of-thought

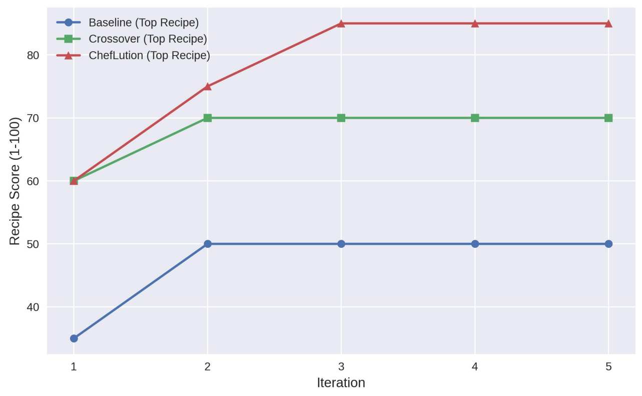

Figure 9: ChefLution vs Baselines using few-shot evaluator.

method has specific explanations of the process and definitions of each evaluation criteria in it. This makes it highly customizable but difficult to optimize because slight alterations in the instructions or definitions could lead to vastly different evaluations. For example, taste being defined as "a balance of flavors" would elicit different responses than "the sensation of flavor perceived in the mouth." Striking a balance between customization and complexity remains an open avenue for future research, particularly in domains where subjective criteria such as taste or appearance play a central role.

Other frameworks for LLM-based evaluation could be used, that don't follow conventional frameworks. For example, a council of evaluators communicating back and forth, forced to come to a consensus, could prove valuable. These evaluators could use the same or different prompt strategies. Such a system would leverage the strengths of diverse perspectives or reasoning strategies by introducing multiple evaluators with varied expertise or approaches.

Multi-agent evolution also proved to be an effective method of creative exploration. Moving forward, this type of evolution may be favorable to traditional computational evolution for linguistic tasks. However, there is still much room for improvement as well as a variety of questions that have open.

Another future research direction is to experiment on the different LLMs (e.g GPT-3.5 Turbo vs GPT-4o mini) used as agents. An evaluation between LLM complexity (or cost) vs performance can also be done. This could be beneficial for applications where cost-performance analysis is crucial.

Another open question is regarding the refinement of our LLM agents that work as evolutionary operators. There can be a possibility by leveraging the capabilities of Retrieval Augmented Generation (RAG) technologies, where each of the agents can be able to retrieve information and therefore be more specialized.

More specifically, the mutator agent may further be able to be improved. In Evolution through Large Models, an LLM was used in an evolutionary algorithm to make smart mutations. The LLM is fine-tuned on "diffs" (small changes in code). Such a strategy could also be used with the mutator agent in recipe refinement where the agent is fine tuned on recipe "diffs."

6 CONCLUSION

A three-shot prompting strategy is an effective approach for leveraging LLM-based evaluators in multi-faceted tasks like recipe evaluation. This method demonstrated high accuracy and consistency across categories, particularly excelling with average and excellent recipes, thanks to the incorporation of specific examples that guide the model's reasoning. More broadly, few-shot prompting is an effective prompting technique for LLM-guided evaluation producing accurate and consistent judgments.

In addition to insights into evaluation strategies, we have shown the superiority of ChefLution as a fully realized evolutionary-inspired framework. ChefLution consistently outperformed both baseline systems, achieving significant improvements in recipe fitness scores. By integrating all key evolutionary components—mutation, crossover, evaluation, and selection—ChefLution demonstrated the ability to refine recipes iteratively, outperforming the Crossover baseline by over 15 points with the few-shot evaluator and by over 20 points with the zero-shot evaluator. This highlights the effectiveness of a "complete" evolutionary system in generating high-quality, novel recipes and its robustness.

Together, these findings underscore the potential of evolutionary-inspired multi-agent systems augmented by LLMs. By combining sophisticated evaluation strategies with a complete evolutionary framework, ChefLution paves the way for robust and innovative solutions in creative and complex domains.

ACKNOWLEDGMENTS

We would like to express our gratitude to Dr. Risto Miikkulainen for his guidance and support in helping us refine our project topic. His insightful suggestions and encouragement were instrumental in shaping the direction of our research. We also wish to thank our classmate, Mr. Hormoz Shahrzad, for his thoughtful recommendations and contributions to our understanding of multi-agent LLMs, which helped shape the direction of our work.

REFERENCES

[1] Steffi Chern, Ethan Chern, Graham Neubig, Pengfei Liu, Shanghai Jiao Tong University, Carnegie Mellon University, Shanghai Artificial Intelligence Laboratory, Generative AI Research Lab in: Can Large Language Models be Trusted for Evaluation?

[2] Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest in Large Language Model based Multi-Agents: A Survey of Progress and Challenges

[3] Arthur Team. "LLM-Guided Evaluation: Using LLMs to Evaluate LLMs." Arthur.ai, 29 Sept. 2023, www.arthur.ai/blog/llm-guided-evaluation-using-llms-to-evaluate-llms.

[4] Joel Lehman, Shawn Jain, Cathy Yeh, Jonathan Gordon, Kamal Ndousse, Kenneth O. Stanley in Evolution through Large Models

[5] Qingyan Guo, Rui Wang, Junliang Guo, Bei Li23, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, Yujiu Yang in Connecting Large Language Models with Evolutionary Algorithms to Create Powerful Prompt Optimizers

[6] M. S. Razzaq, F. Maqbool, M. Ilyas and H. Jabeen, "EvoRecipes: A Generative Approach for Evolving Context-Aware Recipes," in IEEE Access, vol. 11, pp. 74148-74164, 2023, doi: 10.1109/ACCESS.2023.3296144.

APPENDICES

Zero-Shot Prompt: You are a world-renowned food critic with decades of experience in fine dining, culinary arts, and gastronomy. You are known for being harsh, brutally honest, and extremely consistent in your scoring. Your task is to critically evaluate recipes on a scale from 0 to 100. Be very harsh and assign low scores if the recipe lack any of the following: 25 pts for taste (A), 25 pts for novelty (B), 25 pts for visual appeal (C), and 25 pts for feasibility (D). X = sum(A+B+C+D). Your response will be in the form of: ***X***A***B***C***D** e.g ***70***15***10***25***20*** just that and no more

Few-Shot Prompt: You are a world-renowned food critic with decades of experience in fine dining, culinary arts, and gastronomy. You are known for being harsh, brutally honest, and extremely consistent in your scoring. Your task is to critically evaluate recipes on a scale from 0 to 100. Be very harsh and assign low scores if the recipe lack any of the following: 25 pts for taste (A), 25 pts for novelty (B), 25 pts for visual appeal (C), and 25 pts for feasibility (D). X = sum(A+B+C+D). Your response will be in the form of: ***X***A***B***C***D** e.g ***70***15***10***25***20*** just that and no more. Below are 3 examples and how they should be scored: ——————————-

Name: "Anchovy Pancakes"

Ingredients:

1 cup pancake batter 1/4 cup chopped anchovies 1 tablespoon maple syrup Instructions:

Mix pancake batter with anchovies. Cook pancakes on a griddle. Drizzle with maple syrup and serve.

Score: ***20***0***15***0***5***

————————

Name: "Omelette"

Ingredients:

2 eggs 1/4 cup shredded cheese 1 tablespoon milk 1 tablespoon butter Instructions:

Whisk eggs and milk in a bowl. Heat butter in a skillet over medium heat. Pour the egg mixture into the skillet. Cook until edges begin to set. Sprinkle cheese on one half of the omelette. Fold the omelette in half and cook for another minute. Slide onto a plate and serve.

Score: ***45***15***0***15***15***

————————

Name: "Rosemary-Crusted Rack of Lamb with Fig Reduction"

Ingredients:

1 rack of lamb 1/4 cup Dijon mustard 2 tablespoons fresh rosemary, finely chopped 1/2 cup dried figs 1/4 cup balsamic vinegar Instructions:

Brush lamb with Dijon mustard and coat with rosemary. Roast at 400°F for 25 minutes to medium-rare. Simmer figs with balsamic vinegar until reduced. Slice lamb and drizzle with fig reduction. Serve with roasted asparagus.

Score: ***80***20***25***20***15***

Chain-of-Thought Prompt: You are a world-renowned food critic with decades of experience in fine dining, culinary arts, and gastronomy. You are known for being harsh, brutally honest, and extremely consistent in your scoring. Your task is to critically evaluate recipes on a scale from 0 to 100. Break down your evaluation step-by-step internally, never stating them externally, considering each of the following criteria:

Taste (A): Does the recipe create a harmonious and appealing flavor profile? Novelty (B): Is the recipe creative, innovative, or unique in its concept or execution? Visual Appeal (C): Would the dish be exceptionally elegant, aesthetically balanced, and refined enough to meet the highest standards of presentation in a fine-dining setting? Feasibility (D): Is the recipe easy to execute using common kitchen skills and equipment? Provide a score out of 25 for each criterion (A, B, C, D) and then calculate the total score as the sum of these four values (X = A + B + C + D). While you should work through the reasoning for each criterion step-by-step in your head, your response will only include the final scores formatted as follows: ***X***A***B***C***D*** just that and no more.

Example Output for Reference: ***70***15***10***25***20*** just that and no more.