Cooking Up Creativity: Enhancing LLM Creativity through Structured Recombination

Moran Mizrahi[1] **Chen Shani**[2] **Gabriel Stanovsky**[1]
Dan Jurafsky[2] **Dafna Shahaf**[1]

[1]The Hebrew University of Jerusalem, Israel [2]Stanford University, USA
{moranmiz, gabis, dshahaf}@cs.huji.ac.il
{cshani, jurafsky}@stanford.edu

Abstract

Large Language Models (LLMs) excel at many tasks, yet they struggle to produce truly creative, diverse ideas. In this paper, we introduce a novel approach that enhances LLM creativity. We apply LLMs for translating between natural language and structured representations, and perform the core creative leap via cognitively inspired manipulations on these representations. Our notion of creativity goes beyond superficial token-level variations; rather, we recombine structured representations of existing ideas, enabling our system to effectively explore a more abstract landscape of ideas. We demonstrate our approach in the culinary domain with *Dish*COVER, a model that generates creative recipes. Experiments and domain-expert evaluations reveal that our outputs, which are mostly coherent and feasible, significantly surpass GPT-4o in terms of novelty and diversity, thus outperforming it in creative generation. We hope our work inspires further research into structured creativity in AI.

1 Introduction

Large Language Models (LLMs) excel at generating fluent, coherent text and performing tasks that draw on extensive world knowledge. However, they often struggle to generate truly creative ideas (Franceschelli and Musolesi, 2024; Chakrabarty et al., 2024a; Tian et al., 2024b; Zhao et al., 2025).

In creativity research, creative outputs are typically defined as those that are both ***novel*** (unexpected and original) and ***valuable*** (useful, relevant, or effective) (Mumford, 2003; Boden, 2004). However, due to LLMs relying on vast repositories of existing data, they inherently follow learned patterns, making them prone to producing **predictable, repetitive** outputs that lack genuine novelty. Ironically, attempts to explicitly instruct LLMs to ‘‘think more creatively’’ often lead them to generate **invalid or hallucinated** (i.e., invaluable) solutions that could mislead uninformed users (Wang et al., 2024a; Jiang et al., 2024). Together, these limitations make creative generation a persistent challenge for LLMs.

The *temperature* parameter of LLMs controls the amount of randomness, and is often claimed to be the creativity parameter, i.e., the implicit way to enhance creativity in LLMs. However, creativity encompasses much more than mere randomness; a recent study (Peeperkorn et al., 2024) found that while higher temperatures weakly correlate with increased novelty, their actual influence on overall creativity remains subtle and limited.

Much recent work has shown that combining LLMs with structured knowledge (e.g., knowledge graphs) can significantly improve their performance, especially in inference and reasoning tasks (Feng et al., 2023; Sun et al., 2023; Pan et al., 2024; Wang et al., 2024b). Several works use LLMs to parse text into structured representations, manipulate these representations, and (optionally) apply the LLM again to translate the result into text (Yang et al., 2023; Zelikman et al., 2023; Besta et al., 2024; Zhang et al., 2025).

In this work, we show that surprisingly, incorporating structure can also improve LLMs’ *creativity* and *diversity*. We stress that we do not mean creativity and diversity on the lexical (token) level; rather, we want the model to be creative on a more abstract level, within the realm of concepts (or the ‘‘landscape of ideas’’, so to speak).

Our approach is illustrated in Figure 1. Similar to parsing-based approaches, we start by deriving structured representations from textual inputs using an LLM. We then manipulate these structured representations externally, thereby generating new ideas while systematically exploring

Transactions of the Association for Computational Linguistics, vol. 14, pp. 418–441, 2026. https://doi.org/10.1162/TACL.a.638
Action Editor: Trevor Cohn. Submission batch: 3/2025; Revision batch: 7/2025; Published 4/2026.

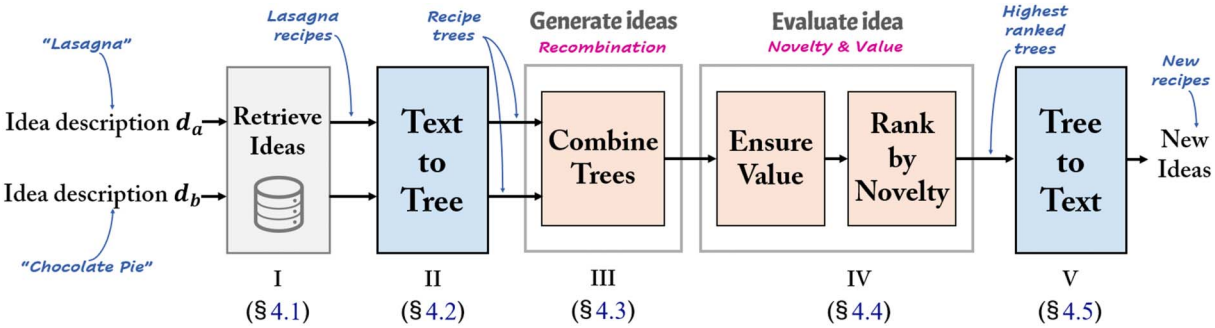

Figure 1: The *Dish*Cover pipeline for creative recipe generation (with LLM-based components shaded in blue) takes as input two idea descriptions. Each description is mapped to a set of specific recipes (§ 4.1), which are parsed into tree representations (§ 4.2). These trees are subsequently combined using a minimal edit distance algorithm (§ 4.3), assessed for value, and ranked based on novelty scores (§ 4.4). Finally, the highest-ranked trees are translated back into natural language recipes (§ 4.5).

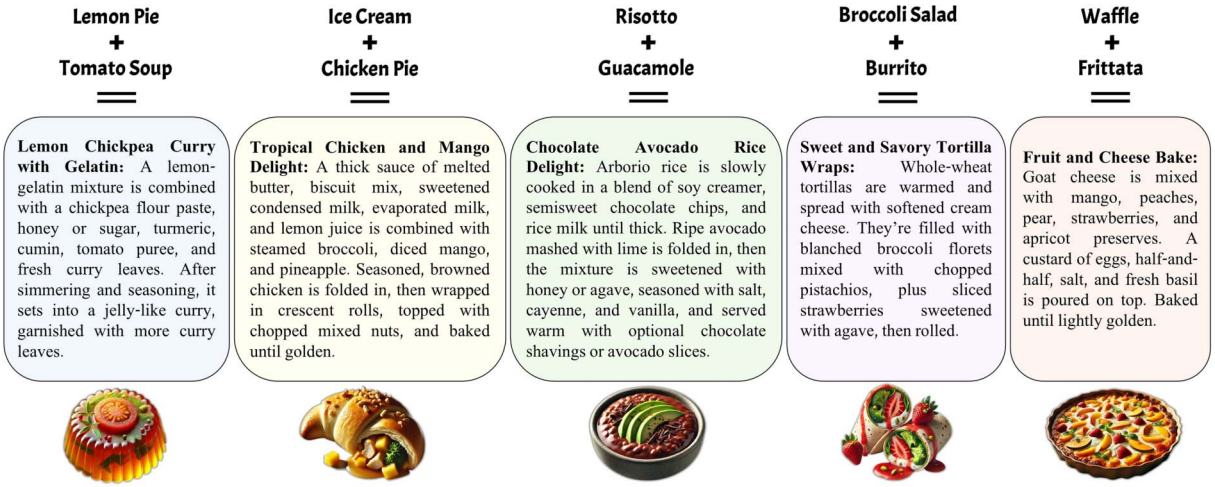

Figure 2: Examples of new recipe ideas generated by *Dish*Cover. Each example consists of a pair of input dishes and their best corresponding generated recipe idea. The generated recipes are presented as concise summaries to conserve space. Images were generated using OpenAI's DALL·E.

creative regions of the idea space. Inspired by the human creative process, we focus on **recombination**—a fundamental principle in creativity research, which posits that novel ideas often emerge by merging existing concepts in unexpected ways (Guilford, 1967; Utterback, 1996; Ahuja and Morris Lampert, 2001). For example, combining pizza preparation methods with the flavors of alfajores cookies might yield a brand new ''alfajores pizza''; combining a sofa with a bookshelf might result in multifunctional furniture. To recombine structured representations of existing ideas, we employ an edit-distance algorithm, and focus on representations midway through its transformation steps. We sample from the space of recombinations, evaluating candidates for novelty and value (Finke et al., 1996; Sawyer and Henriksen, 2024). Those deemed most promising are then translated into natural language.

We demonstrate our paradigm in the culinary domain, introducing *Dish*Cover, a model for creative generation of recipes. Figure 2 presents examples of recombinations generated by it. Beyond the scope of cooking, we believe this paradigm holds promise for extending creative and diverse generation to domains where suitable structured representations and value criteria can be defined (e.g., procedural texts, drug design, music generation; see Section 8). Our contributions are:

- We introduce a novel paradigm to enhance LLM creativity by extracting structured representations, applying cognitive inspired manipulations, and decoding the results back

into natural language, thus going beyond superficial token-level variation.

- We propose a new recombination operator based on edit distance, which enables controlled blending of structured ideas by partially transforming one representation into another.
- We demonstrate our approach in the culinary domain with *Dish*COVER, a model that recombines recipes to generate creative ones.
- We curate a 5K-recipe dataset generated by *Dish*COVER, providing a valuable resource for future work on creative generation. We make both the code and data publicly available.[1]
- Through systematic experiments, we show that *Dish*COVER's generations are significantly more **diverse** compared to baseline SOTA LLM outputs. Most recipes generated by both models are deemed valuable (appropriate and coherent), although the baseline achieves better scores on an open-ended task. Most importantly, our outputs significantly surpass the baseline in terms of novelty, resulting in more **creative** culinary ideas. These findings are supported by both automated metrics and domain expert evaluations.

2 Background: Human Creativity

The field of human creativity has been extensively studied, identifying numerous principles that drive innovation. In designing our model, we relied on the following principles:

Generation & Evaluation. A common yet effective model of creative thinking is the two-stage process *generation & evaluation*, which suggests that creativity begins with divergent thinking (free idea generation), followed by convergent thinking, where the most promising ideas are selected and refined (Finke et al., 1996; Sawyer and Henriksen, 2024). We incorporate this as the conceptual backbone of our model, implementing a generative component that produces a broad set of ideas, followed by an evaluative component that identifies those with the greatest creative potential.

Recombination of Ideas. We base our work on a prominent idea-generation method: *recombination*, where elements from existing ideas are merged to create novel concepts (Koestler, 1964; Guilford, 1967). Our model strategically recombines elements from pairs of existing seed ideas to spark unexpected connections.

Creativity Assessment: Novelty & Value. After generating many ideas, the challenge is determining which are genuinely creative. Numerous studies have examined the complexities of assessing creativity in both humans and computational systems (Said-Metwaly et al., 2017; Lamb et al., 2018). A widely accepted definition of creativity frames it as the intersection of *novelty* and *value* (Mumford, 2003; Boden, 2004, 2009; Lamb et al., 2018). Novelty ensures that an idea is surprising or unconventional, while value signifies it is useful in its intended context.

Measuring Novelty and Value Automatically. Novelty can be assessed by identifying how uncommon an idea is within a dataset (Heinen and Johnson, 2018; Kenett, 2019; Doboli et al., 2020). Evaluating value, however, is highly domain-dependent, often considered the ''holy grail'' of computational creativity (Boden, 2004; Ritchie, 2007; Jordanous, 2012). Thus, we consider value assessment as a domain-specific task.

3 Problem Definition

Innovation often involves combining existing ideas to create novel ones. This process, often referred to as ''conceptual blending'' or ''creative recombination,'' is central to innovation, and the focus of our work. We now introduce **key elements** of our formulation.

Given a domain where ideas can be expressed in a structured format (e.g., cooking recipes, instruction manuals, computer programs), let $\mathcal{I}$ denote the theoretical set of all possible ideas within that domain—both existing and yet-to-be-discovered. $\mathcal{I}$ represents the entire conceptual space of ideas that adhere to the domain's structural and logical constraints, encompassing all valid possibilities. In addition, let $I \subset \mathcal{I}$ be a set of ideas that have been recorded or are known within the field.

Definition 1 (**Recombination Function**). *Recombination function $\mathcal{C}$ takes as input two structured ideas $i_a, i_b \in I$ and produces a set of new combinations $I_{ab} \subseteq \mathcal{I}$ such that each $i \in I_{ab}$ is a different mixture of i_a, i_b.*

[1] https://github.com/moranmiz/Cooking-Up-Creativity.

The exact definition of ''mixture'' depends on the representation. For example, when we transition from representation i_a to representation i_b with a minimal edit distance procedure, the intermediate steps can be viewed as mixtures of i_a and i_b, blending elements of both in varying proportions as we move through the transformation.

Definition 2 (Evaluation Function). *The output of a recombination is a set of potential innovations I_{ab}, which can be evaluated with evaluation function $E : \mathcal{I} \rightarrow \mathbb{R}$. The innovation can be evaluated based on criteria such as novelty and utility.*

Definition 3 (Retrieval of Ideas from Descriptions). *Ideas are often expressed in different levels of abstraction and granularity. Let m be a function that matches an idea description d to relevant known ideas from I, $m(d) \subseteq I$. For example, m could match the textual description ''lasagna'' to all lasagna recipes.*

The formal optimization problem can thus be stated as: Given two idea descriptions d_a, d_b, find

$$\underset{i \in \mathcal{C}(i_a, i_b) \mid i_a \in m(d_a),\, i_b \in m(d_b)}{\operatorname{argmax}} E(i)$$

Figure 2 illustrates examples of generated ideas in the domain of cooking recipes, along with the idea descriptions used to create them. For example, combining broccoli salad recipes and burrito recipes resulted in a recipe for a tortilla filled with cheese, broccoli, strawberries, and pistachios.

4 Model

In this section, we introduce *DishCover*, our model for automatically generating innovative recipes.[1] Figure 1 illustrates our methodology. The input consists of two seed inspirations (idea descriptions d_a, d_b).[2] Each idea description is mapped to a set of specific recipes (instances of the idea, for example different lasagna recipes; § 4.1).

These recipes are first translated into tree representations using an LLM (step (I) in Figure 1, § 4.2). We recombine these trees to produce new candidate ideas with a minimal edit-distance algorithm (step (II), § 4.3). Then, we refine the candidate ideas to assess their value and rank them based on their novelty scores (step (III), § 4.4). Finally, the highest-ranked trees are translated back into natural language recipes using an LLM, which leverages its commonsense and world knowledge to fill in missing details and produce coherent recipes (step (IV), § 4.5). We now provide more details about each step.

4.1 Sampling Seed Ideas (Step I)

We selected the 100 most popular dishes (e.g., chicken salad, cheesecake) that span different categories (e.g., appetizers, desserts, main courses) from the Recipe1M+ dataset (Marın et al., 2021). On average, each selected dish is associated with 2,576.33 recipes in the dataset.

To keep the financial costs of using an LLM manageable in the next stage of our model, we sampled 30 recipes per dish, resulting in a total of 3K recipe samples. To ensure both diversity and representativeness, we selected 15 recipes at random to capture the typical version of each dish, and 15 more to maximize diversity. Implementation of diversification can depend on the representation and the domain; in our case, we use the Gaussian Mixture Model algorithm over recipe embeddings obtained from a Sentence-BERT model fine-tuned on recipe data (see details in Appendix A), which identifies the dish's embedding centroid and iteratively selects recipes farthest from both the centroid and previously chosen samples (Ravi et al., 1994).

4.2 Text to Tree (Step II)

Cooking recipes, like experiments, assembly manuals, and game instructions, are procedural texts. These texts typically consist of a sequence of steps accompanied by the objects needed to perform them. A common way to represent procedural texts is as a tree (Jermsurawong and Habash, 2015; Maeta et al., 2015), where leaf nodes correspond to needed objects (in our case, ingredients), and internal nodes represent the actions performed on them. Figure 3 shows simple lasagna and chocolate pie recipes represented as trees.

To parse recipe text into tree, we prompted GPT-4o with a chain-of-thought approach. See Appendix B & C for full details and corresponding prompts. The total cost of generating tree representations for 3K recipes was approximately $40.

An initial pass revealed that 1,347 (44.9%) of the resulting trees were invalid due to issues such as orphan nodes, multiple outgoing edges from a

[2]Note that more inspirations can be used if desired.

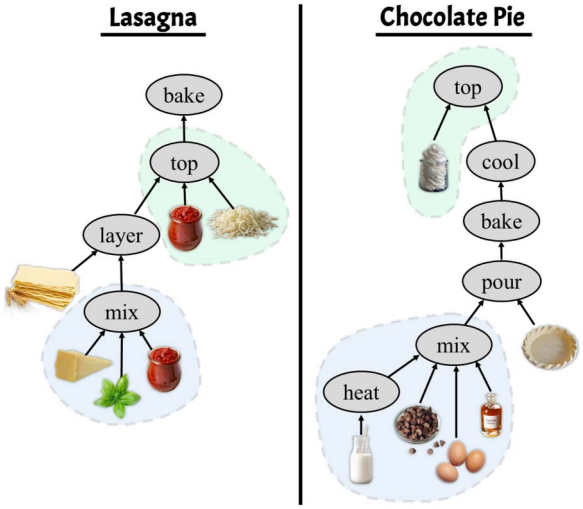

Figure 3: Tree representations of lasagna and chocolate pie recipes. Analogous parts are highlighted, showing structural similarities that the minimal edit distance algorithm is more likely to preserve when transforming one tree into another.

single node, or incorrect edge directions. To address this, we implemented a correction step where we removed problematic edges and instructed the model to reconsider them. This improved validity to 95% (2,850 trees). We then evaluated the final trees using 50 random recipes. A gold-standard tree was created for each recipe, and we automatically compared the predicted trees with the gold trees in terms of node and edge matching. For nodes, we achieved 0.985 precision, 0.956 recall, and F1-Score of 0.969. For edges, we obtained 0.951 precision, 0.909 recall, and F1-Score of 0.93. Overall, these results demonstrate the effectiveness of our approach in translating procedural texts into a structured tree representation, making them suitable for further manipulation and analysis.

4.3 Generate Ideas (Step III)

In this section we operationalize the recombination function $\mathcal{C}$. We generate novel recipes by blending recipe trees with a minimal edit-distance algorithm. The key idea behind this method is that by examining the step-by-step transformation between two concepts, we can discover intermediate forms that blend features of both.

In the case of recipe generation, we employ the Zhang–Shasha algorithm, which computes the minimal edit distance between trees (Zhang and Shasha, 1989; Bille, 2005). Given two recipe trees i_a and i_b, we compute their minimal edit distance and record all operations required to transform i_a into i_b. Each operation sequence produces intermediate trees that represent novel ‘‘merged’’ ideas, from which we randomly select one as our new recipe. Figure 4 illustrates an example sequence transforming a simple lasagna tree into a simple chocolate pie tree. One intermediate variant might be a chocolate lasagna with basil; another could feature a chocolate lasagna encased within a crust.

A key advantage of the minimal edit distance approach is its ability to preserve the structural roles of ingredients and cooking steps. For example, in Figure 3, both the lasagna and chocolate pie recipes include a ‘‘topping’’ action (marked in color). Using a minimal edit distance ensures that inserting whipped cream aside of tomato sauce and cheese is more likely, as placing whipped cream elsewhere would increase the overall edit cost. See implementation details in Appendix D.

Note that stopping at different points in the transformation process can create unique dishes (see Figure 4). Additionally, shuffling the order of edits can generate entirely new intermediate ideas.

4.4 Evaluate Ideas (Step IV)

Now that we have a set of candidate innovations created through recombination, we evaluate each generated recipe. As noted in Section 3, the evaluation function E should take into account novelty and value. Specifically, we chose to view value as a constraint, and novelty as the optimization objective; i.e., we wish to rank by novelty all candidates that pass a value threshold (i.e., make sense).

In the culinary domain, a recipe is considered creative if it introduces unexpected ingredient or technique combinations (novelty) while resulting in a delicious, coherent dish (value, utility). We next operationalize these criteria. Our implementation is specific to recipes, but we believe the principle can be generalized to other domains.

4.4.1 Value Constraint

To assess value (taste) of a recipe, we follow Varshney et al. (2019), and check if its ingredients pair well with each other. We use the flavorDB dataset (Garg et al., 2018), which catalogs taste molecules for raw ingredients. According to this work, two ingredients pair well if they share a larger proportion of taste molecules; we compute a Jaccard-based pairing score for each pair of raw

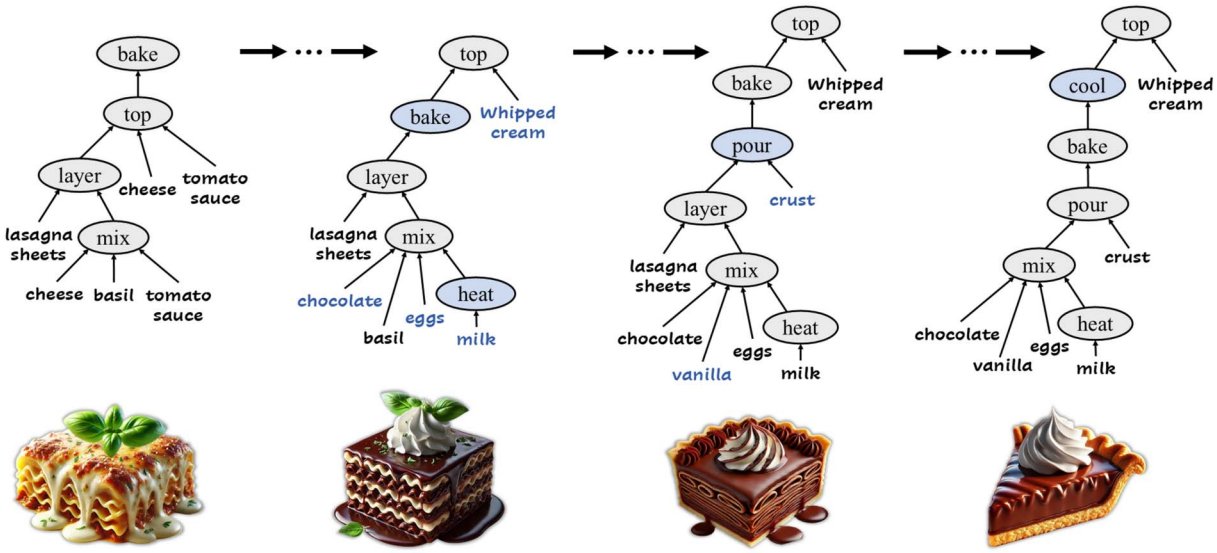

Figure 4: Illustration of our tree-based edit-distance approach for generating new recipe ideas by transforming a simple lasagna tree (left) into a simple chocolate pie tree (right). Intermediate ''merged'' trees are produced for each edit step, yielding novel dishes such as a *basil chocolate lasagna* or a *chocolate lasagna encased in a pie crust*. Recipe images were generated using OpenAI's DALL·E model.

ingredients. Since this dataset does not cover processed ingredients, we also use FoodData Central (Fukagawa et al., 2022) to infer their raw components. We define the pairing score between two composite ingredients as the lowest score among their constituent raw-ingredient pairs. After exploration, we chose 0.3 as our value threshold. If the score falls below this threshold, we consider the pairing problematic. We iteratively remove the ingredient with the highest number of low-scoring pairings until no further collisions remain (alternatively, one might choose to remove these candidates altogether). See details in Appendix E.

4.4.2 Ranking by Novelty

Our intuition is that novel recipes include ingredients and actions that do not often appear together. Thus, we formulate a measure of surprise, inspired by the inverse document frequency (**idf**) concept in **tf-idf** (Ramos, 2003).

Let T be a recipe tree with nodes E_T, where each node represents an **element** (either an **ingredient** or a **cooking action**). Let N_e be the number of recipes in the repository that include element e. For each $e' \in E_T \setminus \{e\}$, let $df_e(e')$ denote the number of recipes containing both e, e'. We define:[3]

$$idf_e(e') = log\Big(\frac{N_e}{df_e(e')}\Big)$$

[3]To prevent score inflation due to typos or extremely rare ingredients, we exclude any ingredient that appears exceptionally infrequently in the entire dataset.

A higher $idf_e(e')$ score indicates that e' is more unique relative to recipes containing e.

To compute novelty of an element e in T, we compute its idf_e score with respect to each of the other elements in T, and sum the top 10 values. To compute the overall novelty of T, we use the sum of the top-10 element-level novelty scores. A chicken and mango delight recipe (Figure 2) was deemed novel due to ingredient combinations like mango, crescent roll, nuts and broccoli, and actions such as steaming, blanching and unrolling.

4.5 Tree to Text (Step V)

After identifying the highest-ranked trees, we convert them back into natural-language recipes using an LLM. Similar to Section 4.2, we employ a chain-of-thought approach, instructing GPT-4o to translate the tree (encoded in DOT format) into a structured recipe, including a title, ingredient list, and step-by-step instructions. We then prompt the model to refine and correct the text for coherence, fluency, and consistency (see Appendix F for full prompts). Processing 1,000 recipe trees in this step required a total cost of approximately $42.

We note that the LLM plays a key role in *surface realization*. Specifically, it draws on commonsense and domain knowledge to (1) **fill in missing details** (e.g., suggesting plausible ingredient quantities or cooking times), and (2) **correct inconsistencies or omissions** introduced during

recombination (e.g., restoring a step for cooking raw chicken if it was omitted).

5 Creative Recipe Dataset

We used our model to generate a new dataset of recipes. We first identified 100 popular dishes spanning different categories. For each dish, we sampled 30 recipes and converted them into tree structures. Next, we sampled 1,000 dish pairs, ensuring that pairs include dishes from different categories. For each pair of dishes, d_a and d_b, every recipe pair in $m(d_a) \times m(d_b)$ was used to generate six blended trees, producing up to 5,600 trees per dish pair ($\sim$5.5M in total). We selected the five highest-ranked trees from each set and converted them back into natural language recipes, resulting in a dataset of 5K recipes.

6 Evaluation

We now turn to evaluate *Dish*COVER by investigating the following research question: ***How do the recipes DishCover generated compare to those generated by a SOTA LLM (GPT-4o)?***

To answer this, we examine how our outputs compare to those of GPT-4o in two key aspects: **diversity** and **creativity**. For diversity, we explore whether our approach mitigates the well-documented issue of repetitiveness in LLMs. For creativity, we look for outputs that are both valuable (make sense) and novel (unexpected).

We compare our model's outputs with those of GPT-4o[4] on two tasks:

Experiment 1. We evaluated GPT-4o and *Dish*COVER on their ability to generate creative recipes **combining a given dish pair**. The evaluation included 10 randomly selected dish pairs. For each pair, we selected 5 recipes generated by GPT-4o and compared them to the top 5 recipes generated by *Dish*COVER (50 recipes per model).

Experiment 2. To broaden the scope of our analysis, we wish to evaluate the most creative recipes GPT-4o and *Dish*COVER could generate in general, **without limiting them to a given input pair**. We used 100 recipes from each model. For *Dish*COVER, we selected 100 recipes from our 5K-recipe dataset, which includes a novelty score for each generated recipe. To ensure that the dishes do not come from a very small number of inputs, we employed simulated annealing (Bertsimas and Tsitsiklis, 1993), maximizing recipe novelty while enforcing a constraint about the maximum number of appearances of each dish.

Note that Experiment 1 is a more unconventional task, one that the model is less likely to have encountered during training. We evaluate the outputs of both experiments using qualitative and automated analyses as well as human annotations.

6.1 Experimental Details

Here we describe the process of generating recipes for the GPT-4o baseline as well as the human evaluation setup.

Recipe Generation for the GPT-4o Baseline. LLMs are known to be sensitive to prompt paraphrases (Sclar et al., 2023; Mizrahi et al., 2024; Voronov et al., 2024). To ensure a fair and competitive baseline, we conducted a thorough prompt design process. First, we constructed large, diverse pools of prompt paraphrases tailored for each experimental task. Specifically, we developed 104 prompts for Experiment 1 and 114 prompts for Experiment 2. These prompts varied systematically along several axes: structure (explicit steps vs. open-ended, explicit chef role vs. no role, etc.), length (concise vs. detailed instructions), creativity-oriented language, and creativity-related constraints.[5]

We tested different temperature settings (ranging from 0.0 to 2.0) for a sample of these prompts. For $t > 1$, GPT-4o often produced gibberish. In line with the findings of Peeperkorn et al. (2024), we observed improvements in novelty at higher temperatures, although they were rather subtle. We therefore selected $t = 1$ as the highest temperature that produced coherent recipes.

To assess the diverse prompt set, we conducted a small evaluation. For Experiment 1, we generated a recipe for each of three sampled dish pairs and every prompt (resulting in 312 recipes). For Experiment 2, we generated three recipes independently per prompt (resulting in 342 recipes).

Qualitative analysis of the generated recipes revealed significant repetition: For experiment 1, GPT-4o consistently generated recipes that prepared each component separately and combined them at the end of the process in the same superficial manner. For example, all burger-waffle

[4]Version: gpt-4o-2024-08-06 (latest stable version of the model at the time of writing this paper).

[5]All prompt variations are publicly available alongside the dataset and code for full reproducibility.

recipes involved cooking the burgers and waffles independently, then simply stacking the burger between two waffles. This observation is supported by high average Self-BLEU scores (BLEU-2 = 0.929, BLEU-3 = 0.877, BLEU-4 = 0.818), and high cosine similarity between recipes of the same pair (average $\sim$0.90 across sampled dish pairs).

For experiment 2, GPT-4o frequently reused known culinary dish concepts with minor variations (e.g., marinated proteins served with quinoa/rice in approximately 60 recipes). This observation is again supported by high self-BLEU scores (BLEU-2 = 0.912, BLEU-3 = 0.829, BLEU-4 = 0.739), and high similarity to the large corpus of existing recipes, with average cosine similarity to nearest corpus neighbor of 0.853 (std = 0.035).

Due to this pronounced repetition, using multiple prompts in the experiments risked redundant outputs, weakening the baseline. Therefore, two culinary experts reviewed the GPT-4o outputs to select the **single best-performing prompt** per experiment—the one that consistently produced the most creative and varied recipes. Importantly, this means that **the GPT-4o baseline was explicitly optimized for the experiments' evaluation criterion** (creativity as judged by human experts), thus giving it a potential advantage over *Dish*COVER.

Finally, we generated the baseline outputs for the experiments using these best-performing prompts within a single chat session, instructing GPT-4o to produce recipes differing from prior outputs to encourage diversity. See the chosen prompts for both experiments in Appendix G.

Human Evaluation Setup. We used Prolific to recruit and manage a total of 48 participants for both experiments. We pre-screened participants based on their cooking experience and frequency, comfort with adjusting recipes, and ability to judge creative outcomes. Participants were paid an estimated hourly rate of £9, adhering to ethical compensation standards. The participants rated randomly selected recipes drawn from the two models on various aspects related to novelty and value (see the full list of questions in Appendix H). To reduce cognitive load, participants initially viewed concise recipe summaries (also generated by GPT-4o), with the option to examine complete recipes if desired. Each recipe was rated by five annotators. Final scores were determined using the median of annotators' ratings, a standard approach for ordinal data to handle outliers. Novelty and value scores were calculated as the mean ratings of their respective questions; the value score was additionally binarized using a threshold of 4 (on a 1–5 scale) across three related questions. On average, participants rated 31.25 recipes (std = 33.827). Participants who consistently completed ratings too quickly (<45 seconds per recipe) were excluded to ensure data quality.

Human Evaluation Reliability. To confirm the robustness of our human evaluation results, given the inherent subjectivity involved in judging creativity, we computed two metrics commonly used in subjective annotation tasks: Interquartile Range (IQR), capturing numeric consistency among annotators, and Krippendorff's alpha, assessing overall agreement reliability.

- **Interquartile Range (IQR):** The average IQR across all questions and recipes was approximately 1.0 (Exp. 1: mean = 0.992, std = 0.33; Exp. 2: mean = 1.008, std = 0.372). About 79% of items in both experiments had an IQR$\leq$1, and only $\sim$3% of items had an IQR$>$2, indicating generally strong numerical consistency. Analyzing questions individually, value-related questions consistently exhibited lower IQRs ($\sim$0.8–0.9), indicating clearer annotator agreement. Novelty-related questions had slightly higher but still acceptable IQRs ($\sim$1.1–1.3), within the expected range for subjective assessments (Margherita et al., 2021; Palomo-Vadillo et al., 2025). This indicated annotators interpreted our guidelines consistently, with typical individual variation for subjective judgments.

- **Krippendorff's alpha:** To further assess inter-rater reliability for ordinal data, we computed Krippendorff's alpha, obtaining values of 0.201 (Experiment 1) and 0.209 (Experiment 2). While modest, these alpha values are consistent with expectations for subjective tasks like creativity evaluation, where individual interpretations naturally differ, even when numerical responses show overall consistency.

6.2 Diversity in Generated Recipes

Diversity plays a pivotal role in creative generation, reflecting how broadly and flexibly new recipes adapt and transform original concepts. In this part of our evaluation, we use both qualitative and automated analyses to compare *DishCover*'s outputs with those generated by GPT-4o, examining how each model integrates concepts from different dishes and their level of repetitiveness.

Deep vs. Shallow Merging of Recipe Concepts. Both *DishCover* and GPT-4o demonstrated the ability to merge ideas from different dishes, but we noticed their outputs exhibited different types of integration. *DishCover* consistently produced more cohesive integrations, where ingredients from both source dishes were woven together into a single, unified recipe. In contrast, GPT-4o tended toward ''shallow merges'', typically preparing each dish separately before combining them at the end. For example, when asked to create a hybrid of muffins and orange salad, GPT-4o invariably proposed baking muffins, preparing an orange salad, and then serving them together, placing the salad on top or on the side. In contrast, *DishCover* generated recipes like a mixed vegetable bake incorporating elements from both dishes, a citrus cake with an orange-based sauce, and a salad that integrated muffin ingredients.

This difference was also reflected in the average ingredient and word counts: GPT-4o's recipes were nearly twice as long and complex as *DishCover*'s. In Exp. 1, *DishCover* had on average 12.3 ingredients (std = 3.71) and 240.28 words (std = 47.98), compared to GPT-4o's 24.98 ingredients (std = 5.2) and 465.84 words (std = 64.8). In Exp. 2, *DishCover* used 13.32 ingredients (std = 4.39) and 264.15 words (std = 51.38) on average, compared to GPT-4o's 23.79 ingredients (std = 3.25) and 371.7 words (std = 42.92). For reference, recipes in the Recipe1M+ corpus averaged only 9.33 ingredients (std = 4.31) and 168.53 words (std = 104.79). This suggests that GPT-4o tends to generate separate sub-dishes, each with its own ingredients and instructions, and then combine them into a final dish rather than blending the culinary ideas.

Fixation on Structures and Ingredients in GPT-4o's Outputs. GPT-4o showed a strong fixation on certain structures and ingredients.

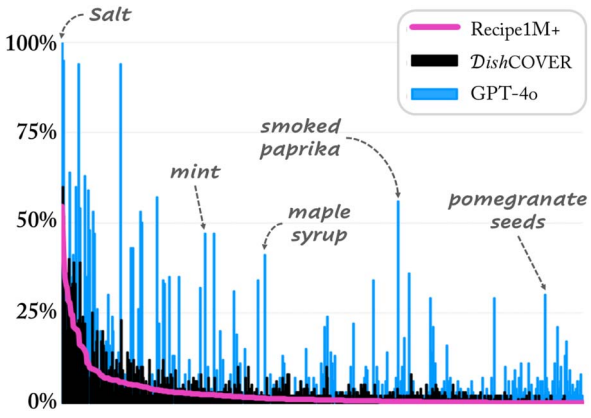

Figure 5: Comparing ingredient frequencies in the original recipe repository (in pink) and the models' generated recipes (Exp. 2). The black histogram shows *DishCover* closely follows the repository's distribution. GPT-4o (blue) shows spikes, highlighting bias to certain ingredients.

When asked to merge two dishes, it often followed the same approach in multiple attempts. For instance, when asked to fuse ''lentil soup'' and ''jam'', GPT-4o repeatedly generated variations of lentil soup served alongside a separately prepared jam, plated in the same way (e.g., placing a dollop of jam in the center of the soup). This fixation persisted when asked to combine specific given recipes, and even when asked to merge broader dish types (e.g., a general soup with a general dip).

Similarly, when asked to suggest a general creative recipe, GPT-4o repeatedly used the same (uncommon) ingredients: 56% of its recipes included smoked paprika (compared to just 0.375% in the repository), 47% used mint, 41% maple syrup, and 29% pomegranate molasses. Figure 5 shows ingredient frequencies in recipes of GPT-4o and *DishCover* against the repository baseline. While *DishCover*'s distribution aligns closely with the original data, GPT-4o has a strong bias toward certain low-frequency ingredients.

Quantifying Diversity via Tree Distances. We computed the average normalized tree edit distance (Rico-Juan and Micó, 2003) between generated recipes after converting them into hierarchical tree structures. In both experiments, GPT-4o's outputs had significantly lower tree distances than *DishCover*'s, indicating higher similarity. In the first experiment, we calculated the average edit distance for each dish pair separately, finding that *DishCover* had an average tree

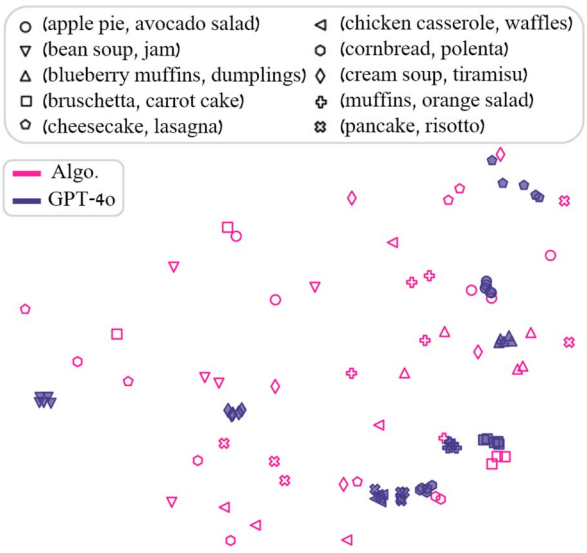

Figure 6: t-SNE visualization for recipe embeddings, experiment 1. Shapes denote dish pairs, colors and fill style denote models. The filled clusters (GPT-4o) of the same shape appear tight and localized, while the unfilled markers (*DishCover*) are scattered, showing higher diversity across *DishCover*'s outputs.

distance of 132.14, while GPT-4o's was 89.35 (p-value = 3.6e-05, paired t-test). In the second experiment, across all outputs, *DishCover* again exhibited greater diversity, with an average tree distance of 140.25 compared to GPT-4o's 129.55 (p-value < 1e-50, two-sample t-test).

Quantifying Lexical Diversity via Self-BLEU. We computed self-BLEU scores across the generated recipes for both experiments. GPT-4o exhibited significantly higher redundancy, reflected in higher self-BLEU scores: In Experiment 1, GPT-4o's outputs scored 0.726 (BLEU-2), 0.655 (BLEU-3), and 0.599 (BLEU-4), while *DishCover*'s scores were substantially lower: 0.423, 0.281, and 0.192, respectively. Similarly, in Experiment 2 GPT-4o scored 0.902, 0.828, and 0.753, compared to *DishCover*'s 0.778, 0.619, and 0.475. The differences are more pronounced in Experiment 1 due to the smaller reference set (5 recipes per pair, vs. 100 recipes).

Quantifying Intra-Set Diversity. We also analyzed the embeddings of the generated recipes using a fine-tuned Sentence-BERT model specialized for cooking recipes (see Appendix A). Figure 6 presents a t-SNE visualization for the first experiment, where each shape represents a dish pair, and color indicates the model (*DishCover* in pink, GPT-4o in blue). GPT-4o's recipes formed

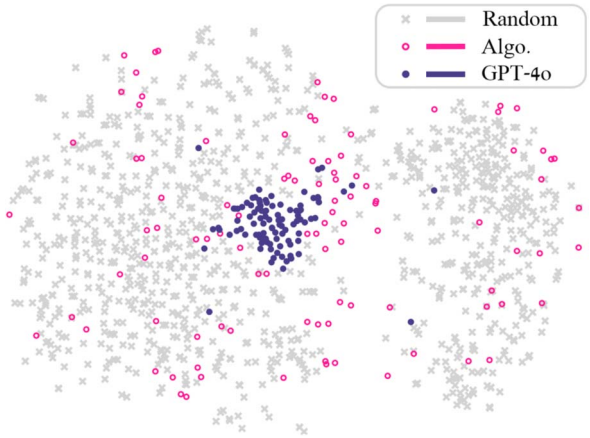

Figure 7: t-SNE visualization for recipe embeddings, experiment 2. The x markers represent 1K random recipes from the general repository. Again, *DishCover*'s outputs are more diverse.

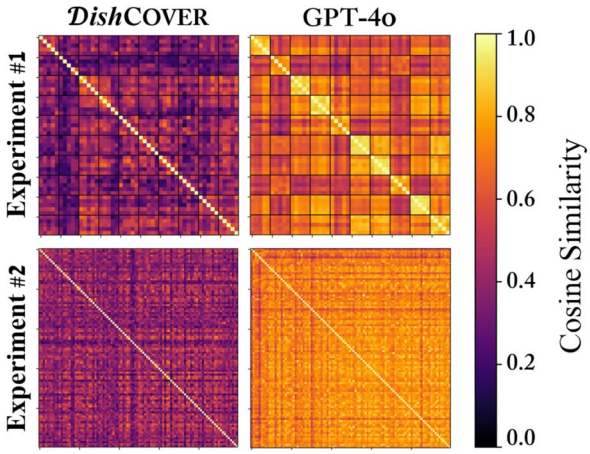

Figure 8: Four heatmaps illustrating pairwise cosine similarities of recipe embeddings for each model across both experiments. GPT-4o's heatmaps (right) display more areas of high similarity, indicating *DishCover*'s higher diversity.

tight clusters for each pair, whereas *DishCover*'s were more dispersed. A similar pattern emerges in experiment 2 (Figure 7). By construction, *DishCover*'s embeddings are widely distributed. The interesting thing to notice, however, is that GPT-4o's embeddings tend to cluster closely. Figure 8 reinforces these findings with heatmaps of cosine similarities among all recipe embeddings. In the first experiment, *DishCover*'s outputs had an average similarity of 0.387 (std = 0.120), while GPT-4o's were higher at 0.659 (std = 0.121). Results were similar in Experiment 2 (*DishCover*: 0.402, std = 0.110; GPT-4o: 0.731, std = 0.078).

Quantifying Diversity with Respect to an External Corpus. We also assessed the similarity of generated recipes to known recipes by computing cosine similarity between a recipe's embedding and that of its nearest neighbor in the Recipe1M+ corpus. *Dish*COVER's recipes consistently exhibited greater distance from existing recipes, with an average similarity of 0.774 (std = 0.061), significantly lower than GPT-4o's average of 0.851 (std = 0.027). Additionally, 91% of *Dish*COVER's recipes had similarity below 0.85, compared to only 47% of GPT-4o's. Notably, 38% of *Dish*COVER's outputs had similarity score below 0.75, while none of GPT-4o's did.

6.3 Human Annotation

While diversity is crucial, genuine creativity in recipe generation also depends on value and novelty. To assess these aspects, we built on the cooking expert annotations from the human evaluation.

In terms of ***value***, both models produced outputs that were largely deemed valuable. In the first experiment, 80% of *Dish*COVER's recipes were classified as valuable, with an average value score of 4.320, compared to 82% for GPT-4o (4.326). In the second experiment, 85% of *Dish*COVER's outputs were classified as valuable (4.36), compared to 98% for GPT-4o (4.51). This makes sense, as we noted that the second experiment (open-ended prompts) is more similar to the data GPT-4o has encountered during training, and also that it tends to output relatively similar recipes.

For ***novelty***, we considered only the recipes that were classified as valuable; this reflects a use-case of a cook skimming a list of recommended recipes, quickly judging their sensibility, and delving only into those that have potential. In both experiments, *Dish*COVER had significantly outperformed GPT-4o in this regard. In the first experiment, *Dish*COVER's average novelty score was 3.53, compared to GPT-4o's 3.146 (p-value = 0.0009). In the second experiment, *Dish*COVER's average novelty score was 3.612, compared to GPT-4o's 3.141 (p-value = 9.2E-08).

Examining the second experiment's valuable outputs further, *Dish*COVER dominated the top quartile of novelty scores, making up 75.55% of the highest-rated recipes, while GPT-4o prevailed in the lower two quartiles, accounting for 71.74% of the less novel outputs. Moreover, among the 37 valuable recipes with a novelty score of 4 or higher, *Dish*COVER contributed 32, leaving only five from GPT-4o. Figure 2 showcases five *Dish*COVER recipes from this high-novelty set. These results strongly suggest that while both models produce mostly valuable recipes (GPT-4o more so, in the open-ended case), *Dish*COVER has a clear advantage in generating truly creative ones.

7 Related Work

Our approach builds on recent parsing-based methods that guide LLMs to map natural language into structured forms, boosting LLM performance in tasks such as constituency parsing (Tian et al., 2024a) and information extraction (Zhao et al., 2023; Li et al., 2024). Similar to research integrating LLMs with Knowledge Graphs (KGs) to improve inference and reasoning (Feng et al., 2023; Jiang et al., 2023; Sun et al., 2023; Wang et al., 2024b), we incorporate domain-specific knowledge into structured representations to provide LLMs with clearer, context-rich signals.

Our work also aligns with models that parse text into structured knowledge, manipulate these representations, and (optionally) convert them into natural language to enhance LLM capabilities (Yang et al., 2023; Zelikman et al., 2023; Besta et al., 2024; Zhang et al., 2025). We surprisingly show that similar techniques can augment creativity and diversity. In parallel, the structured representations we use share a conceptual similarity with latent variable models, which learn to encode inputs into structured latent forms and apply transformations to generate viable alternatives (Kusner et al., 2017; Dai et al., 2018; Zhang et al., 2019). Unlike these approaches, our method does not learn a latent space and requires no training.

Our focus on increasing LLM output diversity complements efforts that promote variation in generation via human feedback (Chung et al., 2023), in-context learning (Zhang et al., 2024), or KG-based interventions (Liu et al., 2021; Hwang et al., 2023b; Liu et al., 2023). Additionally, our work aligns with broader research aimed at fostering creativity in LLMs. One line of work leverages hallucinations for new ideas (Jiang et al., 2024; Yuan and Färber, 2025). Another one draws on insights from human creativity research, incorporating techniques such as constraints (Lu et al., 2025), associative thinking (Mehrotra et al., 2024), role-playing (Chen et al., 2024), and brainstorming

(Summers-Stay et al., 2023; Chang and Li, 2025; Rana and Cheok, 2025). Similarly, we focus on recombination.

Common computational approaches for recombining ideas include conceptual blending frameworks, where merging is guided by heuristics or rules (Fauconnier and Turner, 2003; Pereira and Cardoso, 2006; Veale and Cardoso, 2019); genetic algorithms, which use random crossover points (Corne and Bentley, 2001; Cho, 2002; Dennis and Stella, 2011); and latent space blending, which interpolates between learned representations but lacks explicit structure (Sarkar and Cooper, 2021; Yee-King, 2022; Zhou et al., 2025). Our approach differs by recombining structured representations through an edit distance algorithm, thereby providing explicit and controllable blending.

Beyond improving LLMs, researchers have also explored their use as creative aids for writers (Yuan et al., 2022; Mirowski et al., 2023; Chakrabarty et al., 2024b; Wan et al., 2024), visual artists (Ko et al., 2023), and even humorists (Wu et al., 2025). However, while these tools boost users' sense of creativity, their limited diversity may homogenize the ideas produced across different individuals (Anderson et al., 2024).

Recent work has explored the use of LLMs in the culinary domain (Hwang et al., 2023a; Ma et al., 2024; Zhou et al., 2024; Ataguba and Orji, 2025; Thomas et al., 2025), including several early efforts in recipe generation (H. Lee et al., 2020; Antônio et al., 2020; Bień et al., 2020). Prior to the advent of LLMs, computational recipe generation focused primarily on proposing novel ingredient combinations, often neglecting complete cooking instruction generation (Morris et al., 2012; Cromwell et al., 2015; Amorim et al., 2017; Varshney et al., 2019).

8 Discussion & Future Work

Generalizability to Other Domains. Extending our approach beyond the culinary domain presents important considerations. Our pipeline fundamentally relies on (1) meaningful **structured representations**, and (2) the ability to define or approximate **reliable criteria for assessing value**.

As discussed in Section 4, procedural texts naturally exhibit tree-like structures: leaves often correspond to objects, and internal nodes represent operations performed on them. In storytelling, graphs are commonly used to model narrative arcs and plot dynamics (Elson, 2012; Valls-Vargas et al., 2017). In domains such as drug discovery, molecular structures are inherently graph-based, with atoms as nodes and bonds as edges (Ivanciuc and Balaban, 2000). Music also allows structured, hierarchical representations (Good, 2001; Cuthbert and Ariza, 2010). Other domains, such as poetry or product ideation, lack obvious or standardized structured representations, making the applicability of our technique less immediate.

Value assessment poses another challenge. In recipes, we leverage the FlavorDB dataset as a proxy metric for taste. Our method can be adapted to other domains that employ similar domain-specific metrics. For example, in music, one can predict how humans perceive a combination of sounds (based on psychoacoustic consonance heuristics). Domains related to physical construction (furniture, architecture) could apply automated feasibility checks through CAD software. For programming and game design, functionality could similarly be verified automatically through simulation. In domains where value assessments are inherently subjective and difficult to automate, domain expert evaluation could be integrated into the pipeline, enabling hybrid approaches.

Structured Recombination in LLMs: Implications for AI Creativity and Sampling. A core insight of our work is that structured recombination provides a controllable mechanism to introduce meaningful variability at a higher level of abstraction, rather than the lexical, token level.

To date, attempts to enhance creativity in LLMs often focused on increasing token-randomness (e.g., via temperature), which is not sufficient for creativity (Peeperkorn et al., 2024). Structured recombination enables a model to explore the creative space of ideas more deliberately and effectively. Together with modules for assessing value and novelty this can lead to more creative outputs.

Another promising application of structured recombination is in **structured sampling** and search-based generation tasks. Standard sampling methods in LLMs (Fan et al., 2018; Holtzman et al., 2019) often rely on stochastic techniques that operate at the token level (e.g., nucleus or top-k sampling) to introduce variability. In contrast,

we diversify by sampling over abstract representations rather than raw tokens. We introduce a controlled yet flexible way of navigating idea spaces, which can enhance a range of applications that require deliberate exploration of diverse concepts.

9 Conclusions

In this work, we addressed the persistent challenge of generating creative, diverse outputs. We proposed a novel approach that leverages structured representations to enhance creativity. Rather than relying on superficial token-level variation, we perform cognitively inspired manipulations—specifically, recombining structured representations of existing concepts. We demonstrated our paradigm in the culinary domain through *𝒟ish*COVER, a model designed to generate creative and diverse recipes, and empirically showed significant improvements over GPT-4o in terms of creativity. Ultimately, our approach represents a step toward pushing LLMs beyond surface-level variation, opening the path to richer and more controllable creative generation. We hope this work inspires further research into structured creativity in AI, and we invite the community to build upon this paradigm across diverse domains.

Acknowledgments

We thank the reviewers and action editor for their insightful comments. We further thank Dana Aviran and Eitan Stern for developing the Sentence-BERT fine-tuned model on recipe data, and to the members of **Hyadata Lab** and **SLAB** at the Hebrew University of Jerusalem for their thoughtful remarks. This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 852686, SIAM), by the Israeli Ministry of Science and Technology (grant no. 7256), and by the Koret Foundation grant for Smart Cities and Digital Living.

References

Gautam Ahuja and Curba Morris Lampert. 2001. Entrepreneurship in the large corporation: A longitudinal study of how established firms create breakthrough inventions. *Strategic Management Journal*, 22(6–7):521–543. https://doi.org/10.1002/smj.176

Alvaro Amorim, Luís F. W. Góes, Alysson Ribeiro Da Silva, and Celso França. 2017. Creative flavor pairing: Using RDC metric to generate and assess ingredients combination. In *ICCC*, pages 33–40.

Barrett R. Anderson, Jash Hemant Shah, and Max Kreminski. 2024. Homogenization effects of large language models on human creative ideation. In *Proceedings of the 16th Conference on Creativity & Cognition*, pages 413–425. https://doi.org/10.1145/3635636.3656204

Willian Antônio, João Ribeiro Bezerra, Luís Fabrício Wanderley Góes, and Flávia Magalhães Freitas Ferreira. 2020. Creative culinary recipe generation based on statistical language models. *IEEE Access*, 8:146263–146283. https://doi.org/10.1109/ACCESS.2020.3013436

Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. 1998. Proof verification and the hardness of approximation problems. *Journal of the ACM (JACM)*, 45(3):501–555. https://doi.org/10.1145/278298.278306

Grace Ataguba and Rita Orji. 2025. Exploring large language models for personalized recipe generation and weight-loss management. *ACM Transactions on Computing for Healthcare*. https://doi.org/10.1145/3712709

Dimitris Bertsimas and John Tsitsiklis. 1993. Simulated annealing. *Statistical Science*, 8(1):10–15. https://doi.org/10.1214/ss/1177011077

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690. https://doi.org/10.1609/aaai.v38i16.29720

Michał Bień, Michał Gilski, Martyna Maciejewska, Wojciech Taisner, Dawid Wisniewski, and Agnieszka Lawrynowicz. 2020. RecipeNLG: A cooking recipes dataset for semi-structured text generation. *Proceedings of the 13th International Conference*

on Natural Language Generation, pages 22–28. https://doi.org/10.18653/v1/2020.inlg-1.4

Philip Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1-3):217–239. https://doi.org/10.1016/j.tcs.2004.12.030

Margaret A. Boden. 2004. *The Creative Mind: Myths and Mechanisms*. Routledge.

Margaret A. Boden. 2009. Computer models of creativity. *AI Magazine*, 30(3):23–23. https://doi.org/10.1609/aimag.v30i3.2254

Tuhin Chakrabarty, Philippe Laban, Divyansh Agarwal, Smaranda Muresan, and Chien-Sheng Wu. 2024a. Art or artifice? Large language models and the false promise of creativity. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pages 1–34. https://doi.org/10.1145/3613904.3642731

Tuhin Chakrabarty, Vishakh Padmakumar, Faeze Brahman, and Smaranda Muresan. 2024b. Creativity support in the age of large language models: An empirical study involving professional writers. In *Proceedings of the 16th Conference on Creativity & Cognition*, pages 132–155. https://doi.org/10.1145/3635636.3656201

Hung-Fu Chang and Tong Li. 2025. A framework for collaborating a large language model tool in brainstorming for triggering creative thoughts. *Thinking Skills and Creativity*, page 101755. https://doi.org/10.1016/j.tsc.2025.101755

Jing Chen, Xinyu Zhu, Cheng Yang, Chufan Shi, Yadong Xi, Yuxiang Zhang, Junjie Wang, Jiashu Pu, Tian Feng, Yujiu Yang, and Rongsheng Zhang. 2024. HoLLMwood: Unleashing the creativity of large language models in screenwriting via role playing. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8075–8121. https://doi.org/10.18653/v1/2024.findings-emnlp.474

Sung-Bae Cho. 2002. Towards creative evolutionary systems with interactive genetic algorithm. *Applied Intelligence*, 16:129–138. https://doi.org/10.1023/A:1013614519179

John Chung, Ece Kamar, and Saleema Amershi. 2023. Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 575–593. https://doi.org/10.18653/v1/2023.acl-long.34

David W. Corne and Peter J. Bentley. 2001. *Creative Evolutionary Systems*. Elsevier.

Erol Cromwell, Jonah Galeota-Sprung, and Raghuram Ramanujan. 2015. Computational creativity in the culinary arts. In *FLAIRS*, pages 38–42.

Michael Scott Cuthbert and Christopher Ariza. 2010. music21: A toolkit for computer-aided musicology and symbolic music data. In *11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 637–642.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. 2018. Syntax-directed variational autoencoder for structured data. In *International Conference on Learning Representations*.

John L. Dennis and Aldo Stella. 2011. Teaching creativity: The case for/against genetic algorithms as a model of human creativity. *The Open Educational Journal*, 4(1):36–40. https://doi.org/10.2174/1874920801104010036

Simona Doboli, Jared Kenworthy, Paul Paulus, Ali Minai, and Alex Doboli. 2020. A cognitive inspired method for assessing novelty of short-text ideas. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE. https://doi.org/10.1109/IJCNN48605.2020.9206788

David K. Elson. 2012. *Modeling Narrative Discourse*. Columbia University.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898. https://doi.org/10.18653/v1/P18-1082

Gilles Fauconnier and Mark Turner. 2003. Conceptual blending, form and meaning.

Recherches en Communication, 19:57–86. https://doi.org/10.14428/rec.v19i19.48413

Chao Feng, Xinyu Zhang, and Zichu Fei. 2023. Knowledge solver: Teaching LLMs to search for domain knowledge from knowledge graphs. *arXiv preprint arXiv:2309.03118v1*.

Ronald A. Finke, Thomas B. Ward, and Steven M. Smith. 1996. *Creative Cognition: Theory, Research, and Applications*. MIT Press.

Giorgio Franceschelli and Mirco Musolesi. 2024. On the creativity of large language models. *AI & Society*, pages 1–11.

Naomi K. Fukagawa, Kyle McKillop, Pamela R. Pehrsson, Alanna Moshfegh, James Harnly, and John Finley. 2022. USDA's FoodData Central: What is it and why is it needed today? *The American Journal of Clinical Nutrition*, 115(3):619–624. https://doi.org/10.1093/ajcn/nqab397, PubMed: 34893796

Neelansh Garg, Apuroop Sethupathy, Rudraksh Tuwani, Rakhi Nk, Shubham Dokania, Arvind Iyer, Ayushi Gupta, Shubhra Agrawal, Navjot Singh, Shubham Shukla, Kriti Kathuria, Rahul Badhwar, Rakesh Kanji, Anupam Jain, Avneet Kaur, Rashmi Nagpal, and Ganesh Bagler. 2018. FlavorDB: A database of flavor molecules. *Nucleic Acids Research*, 46(D1):D1210–D1216. https://doi.org/10.1093/nar/gkx957, PubMed: 29059383

Michael Good. 2001. MusicXML for notation and analysis. *The Virtual Score: Representation, Retrieval, Restoration*, 12(113–124):160.

Joy Paul Guilford. 1967. *The Nature of Human Intelligence*. McGraw-Hill.

Helena H. Lee, Ke Shu, Palakorn Achananuparp, Philips Kokoh Prasetyo, Yue Liu, Ee-Peng Lim, and Lav R. Varshney. 2020. RecipeGPT: Generative pre-training based cooking recipe generation and evaluation system. In *Companion Proceedings of the Web Conference 2020*, pages 181–184. https://doi.org/10.1145/3366424.3383536

David J. P. Heinen and Dan R. Johnson. 2018. Semantic distance: An automated measure of creativity that is novel and appropriate. *Psychology of Aesthetics, Creativity, and the Arts*, 12(2):144. https://doi.org/10.1037/aca0000125

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Alyssa Hwang, Bryan Li, Zhaoyi Hou, and Dan Roth. 2023a. Large language models as sous chefs: Revising recipes with GPT-3. *arXiv preprint arXiv:2306.13986v1*.

EunJeong Hwang, Veronika Thost, Vered Shwartz, and Tengfei Ma. 2023b. Knowledge graph compression enhances diverse commonsense generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 558–572. https://doi.org/10.18653/v1/2023.emnlp-main.37

Ovidiu Ivanciuc and Alexandru T. Balaban. 2000. The graph description of chemical structures. In *Topological Indices and Related Descriptors in QSAR and QSPR*, pages 69–178. CRC Press. https://doi.org/10.1201/9781482296945-9

Jermsak Jermsurawong and Nizar Habash. 2015. Predicting the structure of cooking recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 781–786. https://doi.org/10.18653/v1/D15-1090

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023. StructGPT: A general framework for large language model to reason over structured data. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9237–9251. https://doi.org/10.18653/v1/2023.emnlp-main.574

Xuhui Jiang, Yuxing Tian, Fengrui Hua, Chengjin Xu, Yuanzhuo Wang, and Jian Guo. 2024. A survey on large language model hallucination via a creativity perspective. *arXiv preprint arXiv:2402.06647v1*.

Anna Katerina Jordanous. 2012. *Evaluating Computational Creativity: A Standardised Procedure for Evaluating Creative Systems and Its Application*. University of Kent (United Kingdom).

Yoed N. Kenett. 2019. What can quantitative measures of semantic distance tell us about creativity? *Current Opinion in Behavioral Sciences*, 27:11–16. https://doi.org/10.1016/j.cobeha.2018.08.010

Hyung-Kwon Ko, Gwanmo Park, Hyeon Jeon, Jaemin Jo, Juho Kim, and Jinwook Seo. 2023. Large-scale text-to-image generation models for visual artists' creative works. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pages 919–933. https://doi.org/10.1145/3581641.3584078

Arthur Koestler. 1964. *The Act of Creation*. London Hutchinson.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. 2017. Grammar variational autoencoder. In *International Conference on Machine Learning*, pages 1945–1954. PMLR.

Carolyn Lamb, Daniel G. Brown, and Charles L. A. Clarke. 2018. Evaluating computational creativity: An interdisciplinary tutorial. *ACM Computing Surveys (CSUR)*, 51(2):1–34. https://doi.org/10.1145/3167476

Yinghao Li, Rampi Ramprasad, and Chao Zhang. 2024. A simple but effective approach to improve structured language model output for information extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5133–5148. https://doi.org/10.18653/v1/2024.findings-emnlp.295

Chenzhengyi Liu, Jie Huang, Kerui Zhu, and Kevin Chen-Chuan Chang. 2023. DimonGen: Diversified generative commonsense reasoning for explaining concept relationships. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4719–4731. https://doi.org/10.18653/v1/2023.acl-long.260

Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2021. KG-BART: Knowledge graph-augmented BART for generative commonsense reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6418–6425. https://doi.org/10.1609/aaai.v35i7.16796

Yining Lu, Dixuan Wang, Tianjian Li, Dongwei Jiang, Sanjeev Khudanpur, Meng Jiang, and Daniel Khashabi. 2025. Benchmarking language model creativity: A case study on code generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2776–2794. https://doi.org/10.18653/v1/2025.naacl-long.141

Peihua Ma, Shawn Tsai, Yiyang He, Xiaoxue Jia, Dongyang Zhen, Ning Yu, Qin Wang, Jaspreet K. C. Ahuja, and Cheng-I Wei. 2024. Large language models in food science: Innovations, applications, and future. *Trends in Food Science & Technology*, page 104488. https://doi.org/10.1016/j.tifs.2024.104488

Hirokuni Maeta, Tetsuro Sasada, and Shinsuke Mori. 2015. A framework for procedural text understanding. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 50–60. https://doi.org/10.18653/v1/W15-2206

Alessandro Margherita, Gianluca Elia, and Mark Klein. 2021. Managing the COVID-19 emergency: A coordination framework to enhance response practices and actions. *Technological Forecasting and Social Change*, 166:120656. https://doi.org/10.1016/j.techfore.2021.120656, PubMed: 33551496

Javier Marın, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. 2021. Recipe1M+: A dataset for learning cross-modal embeddings for cooking recipes and food images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1):187–203. https://doi.org/10.1109/TPAMI.2019.2927476, PubMed: 31295105

Pronita Mehrotra, Aishni Parab, and Sumit Gulwani. 2024. Enhancing creativity in large language models through associative thinking strategies. *arXiv preprint arXiv:2405.06715v1*.

Piotr Mirowski, Kory W. Mathewson, Jaylen Pittman, and Richard Evans. 2023. Co-writing screenplays and theatre scripts with language models: Evaluation by industry professionals. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*,

pages 1–34. https://doi.org/10.1145/3544548.3581225

Moran Mizrahi, Guy Kaplan, Dan Malkin, Rotem Dror, Dafna Shahaf, and Gabriel Stanovsky. 2024. State of what art? A call for multi-prompt llm evaluation. *Transactions of the Association for Computational Linguistics*, 12:933–949. https://doi.org/10.1162/tacl_a_00681

Moran Mizrahi and Dafna Shahaf. 2021. 50 ways to bake a cookie: Mapping the landscape of procedural texts. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1304–1314. https://doi.org/10.1145/3459637.3482405

Richard G. Morris, Scott H. Burton, Paul M. Bodily, and Dan Ventura. 2012. Soup over bean of pure joy: Culinary ruminations of an artificial chef. In *ICCC*, pages 119–125. Citeseer.

Michael D. Mumford. 2003. Where have we been, where are we going? Taking stock in creativity research. *Creativity Research Journal*, 15(2–3):107–120. https://doi.org/10.1080/10400419.2003.9651403

Maite Palomo-Vadillo, Ana-Lucia Ortega-Larrea, María-Julia Bordonado-Bermejo, and Carmen De-Pablos-Heredero. 2025. Developing an index for measuring gender lens investing in organizations: The GLIMETRICS framework. *Frontiers in Psychology*, 16:1534355. https://doi.org/10.3389/fpsyg.2025.1534355, PubMed: 40370397

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*, 36(7):3580–3599. https://doi.org/10.1109/TKDE.2024.3352100

Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. 2024. Is temperature the creativity parameter of large language models? In *ICCC*.

Francisco C. Pereira and Amílcar Cardoso. 2006. Experiments with free concept generation in Divago. *Knowledge-Based Systems*, 19(7):459–470. https://doi.org/10.1016/j.knosys.2006.04.008

Juan Ramos. 2003. Using TF-IDF to determine word relevance in document queries. In *Proceedings of the First Instructional Conference on Machine Learning*, volume 242, pages 29–48. Citeseer.

Sharif Uddin Ahmed Rana and Adrian David Cheok. 2025. Generative innovation: Leveraging the power of large language models for brainstorming. In *The Economics of Talent Management and Human Capital*, pages 175–192. IGI Global. https://doi.org/10.4018/978-1-6684-6641-4.ch011

Sekharipuram S. Ravi, Daniel J. Rosenkrantz, and Giri Kumar Tayi. 1994. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310. https://doi.org/10.1287/opre.42.2.299

Juan Ramón Rico-Juan and Luisa Micó. 2003. Comparison of AESA and LAESA search algorithms using string and tree-edit-distances. *Pattern Recognition Letters*, 24(9–10):1417–1426. https://doi.org/10.1016/S0167-8655(02)00382-3

Graeme Ritchie. 2007. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17:67–99. https://doi.org/10.1007/s11023-007-9066-2

Sameh Said-Metwaly, Wim Van den Noortgate, and Eva Kyndt. 2017. Approaches to measuring creativity: A systematic literature review. *Creativity: Theories - Research - Applications*, 4(2):238–275. https://doi.org/10.1515/ctra-2017-0013

Anurag Sarkar and Seth Cooper. 2021. Generating and blending game levels via quality-diversity in the latent space of a variational autoencoder. In *Proceedings of the 16th International Conference on the Foundations of Digital Games*, pages 1–11. https://doi.org/10.1145/3472538.3472545

Keith R. Sawyer and Danah Henriksen. 2024. *Explaining Creativity: The Science of Human Innovation*. Oxford University Press. https://doi.org/10.1093/oso/9780197747537.001.0001

Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying language models' sensitivity to spurious features in prompt

design or: How I learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*.

Douglas Summers-Stay, Clare R. Voss, and Stephanie M. Lukin. 2023. Brainstorm, then select: A generative language model improves its creativity score. In *The AAAI-23 Workshop on Creative AI Across Modalities*.

Jiashuo Sun, Chengjin Xu, Lumingyuan Tang, Saizhuo Wang, Chen Lin, Yeyun Gong, Lionel Ni, Heung-Yeung Shum, and Jian Guo. 2023. Think-on-graph: Deep and responsible reasoning of large language model on knowledge graph. In *The Twelfth International Conference on Learning Representations*.

Anna Thomas, Adam Yee, Andrew Mayne, Maya B Mathur, Dan Jurafsky, and Kristina Gligorić. 2025. What can large language models do for sustainable food? In *Forty-second International Conference on Machine Learning*.

Yuanhe Tian, Fei Xia, and Yan Song. 2024a. Large language models are no longer shallow parsers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7131–7142. https://doi.org/10.18653/v1/2024.acl-long.384

Yufei Tian, Tenghao Huang, Miri Liu, Derek Jiang, Alexander Spangher, Muhao Chen, Jonathan May, and Nanyun Peng. 2024b. Are large language models capable of generating human-level narratives? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17659–17681. https://doi.org/10.18653/v1/2024.emnlp-main.978

James M. Utterback. 1996. *Mastering the Dynamics of Innovation*. Harvard Business School Press.

Josep Valls-Vargas, Jichen Zhu, and Santiago Ontanón. 2017. Towards automatically extracting story graphs from natural language stories. In *AAAI Workshops*.

Lav R. Varshney, Florian Pinel, Kush R. Varshney, Debarun Bhattacharjya, Angela Schörgendorfer, and Y.-M. Chee. 2019. A big data approach to computational creativity: The curious case of chef watson. *IBM Journal of Research and Development*, 63(1):7–1. https://doi.org/10.1147/JRD.2019.2893905

Tony Veale and Amílcar Cardoso. 2019. *Computational Creativity: The Philosophy and Engineering of Autonomously Creative Systems*. Springer. https://doi.org/10.1007/978-3-319-43610-4

Anton Voronov, Lena Wolf, and Max Ryabinin. 2024. Mind your format: Towards consistent evaluation of in-context learning improvements. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 6287–6310. https://doi.org/10.18653/v1/2024.findings-acl.375

Qian Wan, Siying Hu, Yu Zhang, Piaohong Wang, Bo Wen, and Zhicong Lu. 2024. ''It felt like having a second mind'': Investigating human-AI co-creativity in prewriting with large language models. *Proceedings of the ACM on Human-Computer Interaction*, 8(CSCW1):1–26. https://doi.org/10.1145/3637361

Dawei Wang, Difang Huang, Haipeng Shen, and Brian Uzzi. 2024a. A preliminary, large-scale evaluation of the collaborative potential of human and machine creativity. *PsyArXiv September, 28*.

Yu Wang, Nedim Lipka, Ryan A. Rossi, Alexa Siu, Ruiyi Zhang, and Tyler Derr. 2024b. Knowledge graph prompting for multi-document question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19206–19214. https://doi.org/10.1609/aaai.v38i17.29889

Zhikun Wu, Thomas Weber, and Florian Müller. 2025. One does not simply meme alone: Evaluating co-creativity between LLMs and humans in the generation of humor. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*, pages 1082–1092. https://doi.org/10.1145/3708359.3712094

Zhun Yang, Adam Ishay, and Joohyung Lee. 2023. Coupling large language models with logic programming for robust and general reasoning from text. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5186–5219. https://doi.org/10.18653/v1/2023.findings-acl.321

Matthew Yee-King. 2022. Latent spaces: A creative approach. In *The Language of Creative AI: Practices, Aesthetics and Structures*, pages 137–154. Springer. https://doi.org/10.1007/978-3-031-10960-7_8

Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. 2022. Wordcraft: Story writing with large language models. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*, pages 841–852. https://doi.org/10.1145/3490099.3511105

Shuzhou Yuan and Michael Färber. 2025. Hallucinations can improve large language models in drug discovery. *arXiv preprint arXiv:2501.13824v2*.

Eric Zelikman, Qian Huang, Gabriel Poesia, Noah Goodman, and Nick Haber. 2023. Parsel: Algorithmic reasoning with language models by composing decompositions. *Advances in Neural Information Processing Systems*, 36:31466–31523.

Jiahuan Zhang, Tianheng Wang, Hanqing Wu, Ziyi Huang, Yulong Wu, Dongbai Chen, Linfeng Song, Yue Zhang, Guozheng Rao, and Kaicheng Yu. 2025. SR-LLM: Rethinking the structured representation in large language model. *arXiv preprint arXiv:2502.14352v1*. https://doi.org/10.18653/v1/2025.acl-long.172

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262. https://doi.org/10.1137/0218082

Tianhui Zhang, Bei Peng, and Danushka Bollegala. 2024. Improving diversity of commonsense generation by large language models via in-context learning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9226–9242. https://doi.org/10.18653/v1/2024.findings-emnlp.540

Xinyuan Zhang, Yi Yang, Siyang Yuan, Dinghan Shen, and Lawrence Carin. 2019. Syntax-infused variational autoencoder for text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2069–2078. https://doi.org/10.18653/v1/P19-1199

Bowen Zhao, Changkai Ji, Yuejie Zhang, Wen He, Yingwen Wang, Qing Wang, Rui Feng, and Xiaobo Zhang. 2023. Large language models are complex table parsers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14786–14802. https://doi.org/10.18653/v1/2023.emnlp-main.914

Yunpu Zhao, Rui Zhang, Wenyi Li, and Ling Li. 2025. Assessing and understanding creativity in large language models. *Machine Intelligence Research*, 22(3):417–436. https://doi.org/10.1007/s11633-025-1546-4

Pengfei Zhou, Weiqing Min, Chaoran Fu, Ying Jin, Mingyu Huang, Xiangyang Li, Shuhuan Mei, and Shuqiang Jiang. 2024. Foodsky: A food-oriented large language model that passes the chef and dietetic examination. *arXiv preprint arXiv:2406.10261v1*. https://doi.org/10.2139/ssrn.4972042

Yufan Zhou, Haoyu Shen, and Huan Wang. 2025. FreeBlend: Advancing concept blending with staged feedback-driven interpolation diffusion. *arXiv preprint arXiv:2502.05606v2*.

A Fine-Tuned Sentence-BERT Model

Our initial experiments with the standard sentence-level Sentence-BERT (SBERT) model revealed that it tends to group recipes emphasizing textual instructions while overlooking ingredients. As a result, it fails to distinguish broad categories (e.g., salad vs. soup) and struggles with finer-grained distinctions (e.g., carrot cake vs. cheesecake).

To better handle recipe similarity, we fine-tuned a Sentence-BERT model.[6] Our fine-tuning dataset consisted of 30K pairs of recipes with their new similarity scores, which equally weighted the original Sentence-BERT score and a Ruzicka-based similarity of the two recipes' ingredient lists as computed by Mizrahi and Shahaf (2021). The dataset was divided into three equal subsets: (1) 10K pairs of recipes, each pair representing instances of the same dish, (2) 10K pairs of recipes from different dishes within the same category (e.g., carrot cake and cheesecake), and (3) 10K pairs of recipes from entirely different categories (e.g., a dessert and a salad).

[6] all-distilroberta-v1.

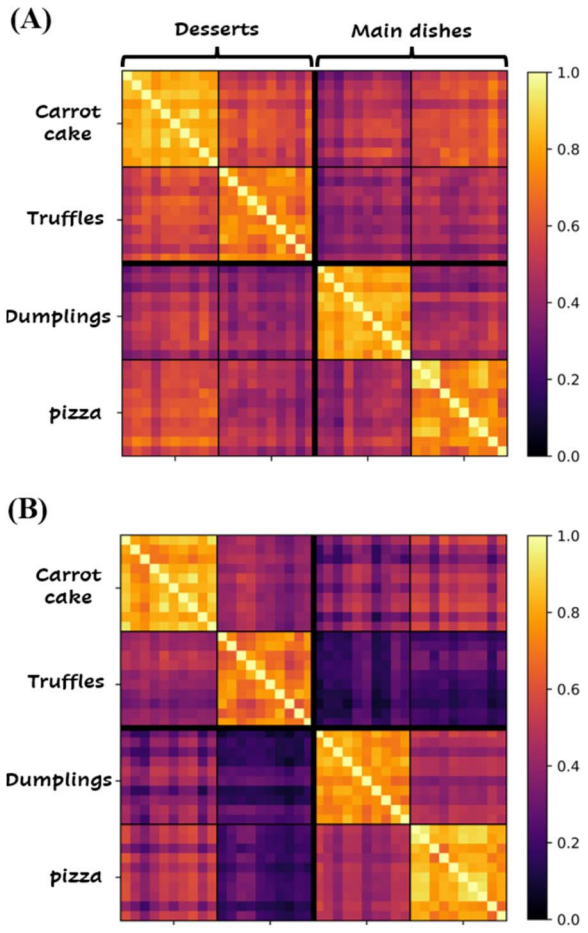

Figure 9: Cosine similarity heatmaps of 40 recipes before and after fine-tuning: (A) Sentence-BERT before fine-tuning, (B) Sentence-BERT after fine-tuning. The original Sentence-BERT model shows uniformly high similarity across all dish pairs, while the fine-tuned model maintains high similarity within the same dish and improves differentiation across categories.

To preserve the original model's capabilities and avoid overfitting, we fine-tuned it for one epoch, reserving 5% of the pairs for validation, achieving a validation accuracy of 92–95%. Following Sentence-BERT fine-tuning guidelines, we employed 10% warm-up steps, used cosine similarity loss as our loss function, and set the maximum sequence length to 512 tokens.

This fine-tuning procedure allowed us to obtain more accurate recipe embeddings that account for both textual instructions and ingredient overlap. Figure 9 presents a cosine similarity heatmap of 40 recipes before and after fine-tuning: 10 highly similar recipes for Carrot Cake, 10 for Truffles (desserts), 10 for Dumplings, and 10 for Pizza (main dishes). As shown, the original Sentence-BERT model exhibits high similarity scores across all dish pairs, including an unexpectedly high similarity between Carrot Cake and Pizza recipes. In contrast, the fine-tuned model retains high similarity scores within the same dish while improving differentiation between dishes within the same category and across categories.

B Text to Tree Parser Details

Parsing the Ingredients. In the first subtask, we instructed the model to parse the ingredients. For each ingredient line, the model extracted (1) the ingredient name, (2) whether the ingredient contributes to the dish's structural core (e.g., lasagna sheets in lasagna) or its flavor (e.g., lemon in lemon pie), and (3) a simplified base form of the ingredient (e.g., ''basil'' $\rightarrow$ ''herb,'' ''walnut'' $\rightarrow$' ''nut''). To evaluate accuracy, we sampled 200 random ingredient lines, and one of us checked whether the parsed information matched the ground truth. We obtained 95% accuracy for ingredient name parsing, 97.5% for the structural vs. flavor reference, and 96% for ingredient base form conversion. These results suggest that GPT-4o performs reliably in parsing ingredients.

Simplifying the Instructions. Next, we asked the model to simplify each sentence in a recipe's instructions while preserving essential content. Specifically, we instructed it to remove details about quantities, sizes, and descriptive elements, ensuring that each simplified sentence contained exactly one action (placed at the beginning) and that ambiguous instructions were converted into active forms (e.g., ''bring to a boil'' $\rightarrow$' ''boil''). For example, the instruction ''Sprinkle salt over the basil and mozzarella and return to the broiler for 1 to 2 minutes, until the cheese is melted and bubbling.'' was simplified into ''Sprinkle salt over basil, mozzarella. Broil until cheese melts''. We tested this phase on 50 sampled recipes, resulting in 451 instruction sentences. One of us reviewed these sentences and found that 423 (93.79%) were simplified correctly. Common errors included using non-informative verbs, omitting a verb or ingredient, or accidentally merging two actions into a single sentence.

Translating the Recipe into a Tree. Finally, we used GPT-4o's coding proficiency to produce a directed tree representation in DOT, a graph description language. We provided the model with a

one-shot example that included a simplified recipe text (dish name, a short list of ingredients, and simplified instructions) along with its corresponding DOT code, annotated with comments to guide the model in structuring the tree representation. To refine the output, we implemented a correction step, automatically removing problematic edges and instructing the model to reconsider them.

C Text to Tree Parser Prompts

This section includes the prompts we used to prompt GPT-4o to parse a recipe text into a tree representation. The system message for all prompts presented here was: ''You are a cooking recipe parser''.

Parsing the Ingredients Prompt: Given a recipe title, id, and ingredients, for each ingredient, determine: (1) Abbreviation: The shortest clear description. (2) Reference Type: Identify if the ingredient is for structure ('structure') or taste ('taste') of the dish. Ingredients impacting both are labeled as 'taste'. (3) Core Ingredient: Boolean indicating if the ingredient is essential to the identity of the dish (e.g., True for chocolate in chocolate cake). (4) Abstraction: Simplify the ingredient to its base form (e.g., 'basil' to 'herb', 'walnuts' to 'nut', 'eggs' to 'egg'). Please return the results in the following JSON format only: {''recipe_id'': [(abbreviation, ref, core, abstraction), ...], ...}. INPUT: recipe_title, recipe_id, ingr_list \n recipe_title, recipe_id, ingr_list \n . . . OUTPUT:

Simplifying the Instructions Prompt: Given the following cooking instructions, please simplify and shorten them as much as possible. Remove quantities, sizes, and descriptions. Ensure each verb initiates a new sentence, and that a sentence does not contain two verbs. Convert permissive or ambiguous instructions into active forms (e.g., ''let cool'' -> ''cool'', ''alternate layers'' -> ''layer''). Return output in JSON format with the key as 'recipe_id' and the value as the full simplified text. INPUT: {recipe_id: <instruction text>, ...} OUTPUT:

Translating the Recipe into a Tree Prompt: Title: . . . Ingredients: . . . Directions: . . . Code: . . . # end of code (the 1-shot example) Title: <dish_name> Ingredients: <ingredient_abbreviation_list> Directions: [i1] <1st_direction> [i2] <2nd_direction> . . . Code:

Tree Correction Prompt: You are provided with the title, ingredients, and directions of a recipe, along with a partial Dot code that represents the recipe's tree structure. The Dot code is missing some edges. Additionally, you will receive names of nodes for which these connections are missing. For each provided node name, add exactly one edge from this node to the action node that uses it (if it is an ingredient) or processes its outcome (if it is an action). Please return only the Dot code for these specific edges, including necessary comments, and exclude any additional text. Title: <dish_name> Ingredients: <ingredient_abbreviation_list> Directions: `[i1]` <1st_direction> `[i2]` <2nd_direction> ... Partial Dot code: <dot_code> Name of nodes with missing edges: <node_names> OUTPUT:

D Tree Edit Distance Implementation

In this appendix, we describe our approach for computing the minimal edit distance between recipe trees. As mentioned in Section 4.3, we employ the Zhang–Shasha algorithm (Zhang and Shasha, 1989), which extends the well-known string edit distance approach to ordered labeled trees. Specifically, a ***labeled tree*** is one in which each node is assigned a symbol from a fixed finite alphabet, and an ***ordered tree*** is one in which each set of siblings has a defined left-to-right order. While computing tree edit distance for unordered trees is NP-hard and even MAX SNP-hard (Arora et al., 1998), the Zhang–Shasha algorithm provides a polynomial-time solution for the ordered case. To make our labeled recipe trees compatible with this approach, we impose an ordering on sibling nodes by sorting them lexicographically according to their labels.

We further adjust edit costs to encourage matching analogous nodes across recipes. Specifically, we allow a node to be substituted by another only if both nodes share the same type (i.e., both are ingredient nodes or both are action nodes). Moreover, a substitution between two nodes incurs zero cost if they have the same label, or a small fixed cost if their labels share the same abstract meaning. For example, two ingredients both categorized as ''herb'' may replace one another at a lower cost than an ''herb'' and a ''liquid'' ingredients. Similarly, two action nodes categorized under ''heat application'' are more likely to substitute for each other than an action node categorized under ''heat

application'' and another one that is categorized under ''flavor enhancement''.

To determine whether two ingredient nodes share the same abstraction, we use the ingredient abstraction obtained from parsing the ingredients (see Appendix B). To determine whether two action nodes share the same abstraction (e.g., ''heat application'' for ''bake'' and ''microwave''), we collected the 250 most common action verbs in recipes and created a hierarchy, grouping these verbs into categories such as heat application, preparation, positioning, flavor enhancement, etc.

Formally, let T_1 and T_2 be two recipe trees composed of ingredient and action nodes. We allow the operations of insertion, deletion, and update. The cost of insertion or deletion is set to 100, whereas the update cost depends on whether the two nodes share the same type and the same label or abstraction. If they have the same type and the same label, the cost is 0; if they share the same type but only the same abstraction, it is 5; otherwise, the cost is ∞, indicating an infeasible substitution. This cost scheme encourages the edit distance algorithm to favor substituting analogous parts over insertions and deletions, resulting in more semantically meaningful transformations.

As noted in Section 4.3, stopping at different points in the transformation process can create unique dishes (see Figure 4). Additionally, shuffling the order of edits can generate entirely new intermediate ideas. To produce more coherent results, we impose a partial order on the shuffled operations. We prioritize inserting and updating key flavor ingredients from the target recipe (e.g., ''lemon'' in a lemon pie) so they appear earlier in the transformation. At the same time, we delay deleting or updating structural ingredients from the source recipe (e.g., ''lasagna sheets'' in lasagna) to preserve its core structure. We determine which ingredients contribute to flavor and which to structure during the parsing phase (see Appendix B). This approach helps maintain the essential characteristics of both dishes, integrating distinct flavor components from the target while retaining the structural integrity of the source. Notably, reversing the transformation (dish B $\rightarrow$ dish A) results in a different edit sequence, leading to distinct new recipes.

To ensure the dishes are indeed recombinations, we discard any recipe that lacks at least one **essential ingredient** from both original dishes (if such an ingredient exists). We define an essential ingredient as one that appears frequently in recipes for a given dish but is not broadly common across all recipes (e.g., lasagna sheets in lasagna). Additionally, we remove ideas that are too similar to the seed recipes. To ensure that the combined tree preserves cross-dish inspiration, we require that at least 30% of its elements (nodes and edges) come from each of the original recipes.

E Identifying Conflicting Ingredients

We remove ingredients that appear to collide in their flavors. Specifically, we look for pairs of ingredients that seldom appear together in the recipe repository, treating their pairing as uncommon and attempt to determine whether it might be a creative success or a failure. To do so, we rely on two external datasets. First, we use **flavorDB** (Garg et al., 2018), which catalogs taste molecules for a wide range of raw ingredients such as fruits, vegetables, and fish. Inspired by this dataset owner's claim that two ingredients pair well if they share a larger proportion of taste molecules, we define a Jaccard-based pairing score between raw ingredients. Since flavorDB does not cover processed ingredients (e.g., lasagna sheets that consist of flour, water, and eggs), we also use **FoodData Central** (Fukagawa et al., 2022) to infer their raw components. We define the pairing score between two composite ingredients as the lowest score among their constituent raw-ingredient pairs. After exploration, we chose 0.3 as our value threshold. If the score falls below 0.3, we consider the pairing problematic.

F Tree to Text Prompts

This section includes the prompts we used to prompt GPT-4o to translate structured tree representation back into natural language. The system message for all prompts presented here was: ''You are a cooking expert''.

Translate Tree into Raw Recipe Prompt: Given the following DOT code, which represents a recipe graphically by defining ingredient nodes, action nodes, and their interconnections, translate the structure into a natural language recipe. The DOT code maps each ingredient to specific actions, and it outlines the order of these actions to demonstrate the cooking process. DOT CODE: '''<recipe_idea_dot_code>''' Convert this

structured representation into a detailed cooking recipe in natural language. Requirements: (1) Output should only include the title, ingredients with quantities, and sequential instructions. (2) Avoid any explanatory comments or embellishments. OUTPUT:

Find Issues and Correct Recipe Prompts: Step I: Review the recipe provided below, which is written in natural language. Identify and list any potential issues with it, excluding any concerns related to unconventional ingredient combinations. Please provide only a list of potential issues without revising the recipe. RECIPE: ‘‘‘<GENERATED RECIPE>’’’

Step II: Please edit the recipe to address the identified issues. Ensure the recipe remains as a single, unified component. Output only the corrected version of the recipe. OUTPUT:

Summarize Recipe Prompt: Please summarize the following recipe in a few sentences: (1) Start with a super concise description of the dish, focusing *only* on its final result. (2) Then, provide a summary of the recipe, including its main components, actions, and all the ingredients used. Use a descriptive tone for this part, avoiding imperative sentences. RECIPE: ‘‘‘<full_recipe>’’’

Review Ingredients Prompt: You are given a description of a creative recipe. CREATIVE RECIPE DESCRIPTION: ‘‘‘<creative_recipe_description>’’’ Your task is to preserve the creative ingredients in the recipe while suggesting the removal or substitution of ingredients that might negatively impact the dish’s flavor. You should: (1) Recognize the unique and unusual ingredients that contribute to the creativity of the dish. (2) Systematically compare all pairs of ingredients in the dish and identify ingredients that have a clear, strong clash with each other due to conflicting flavors. Be thorough and ensure that you include all possible pairs of ingredients that have a strong clash. (3) Based on the identified strong clashes, suggest removals and substitutions of ingredients to avoid clashes, while preserving the creative aspects of the dish. Return only the following JSON output format: {‘‘dish_ingredients’’: <list of strings: the full list of ingredients in the dish>, ‘‘creative_ingrs’’: <list of strings: the list of ingredients that contribute creatively to the dish>, ‘‘flavor_clashes’’: <list of string pairs: the clashing ingredients>, ‘‘removals’’: <list of strings: the list of ingredients to remove>, ‘‘substitutions’’: <list of string pairs: ingredients to substitute - (ingr1, ingr2) means ‘replace ingr1 in ingr2’>}

Increase Readability Prompt: Given the following recipe: (1) Remove the following ingredients: <bad_ingredients>. (2) Make the following ingredient substitutions: <required_substitutions>. (3) Split its ingredients and instructions into distinct sections to improve readability (e.g., ‘‘mix dry ingredients’’, ‘‘assemble’’, etc.). You can change the order of lines but keep the content unchanged. ‘‘‘<full_recipe>’’’

G Chosen Prompts for Experiments

In this section, we present the prompts used to guide GPT-4o in generating recipes for both experiments.

Experiment 1 Prompt: You are a chef at a fusion restaurant that excels in creating delightful and unexpected combinations of classic dishes. Your task today is to design an innovative recipe that merges the intricate layers of {dish1} with the rich decadence of {dish2}. Develop a comprehensive recipe that includes: (1) A unique name that embodies the essence of this fusion dish. (2) A detailed list of ingredients. (3) Step-by-step cooking and assembly instructions, highlighting inventive cooking techniques or unusual ingredient interactions. Promote bold experimentation with flavors and textures to create a dish that is both surprising and satisfying.

Following prompt: Design another different innovative recipe that merges the intricate layers of {dish1} with the rich decadence of {dish2}.

Experiment 2 Prompt: What is the most creative and out-of-the-box recipe you can create?

Following prompt: Create a fresh, unique recipe that differs from the previous ones but matches their level of creativity.

H Human Experiment Questionnaire

1. Do the instructions in this recipe make sense? A recipe that doesn’t make a lot of sense contains technical issues. Issues could be minor (for example, mixing an already-mixed salad) or major (not cooking raw chicken). {scale

(1–5): 1: Throw away the recipe, too many changes needed, 2: Need to change most of the recipe, 3: Requires a lot of changes, 4: Almost perfect, requires some minor changes, 5: Makes perfect sense, could cook it as is}

2. Do the combination of ingredients in this recipe make sense? {scale: same}
3. How similar is this recipe to others you have seen or used? {scale (1–5): 1: Very similar to many recipes, 2: Similar but not very common, 3: Somewhat similar to others, 4: Quite different from most recipes, 5: Very different, highly unusual}
4. How novel is the way the instructions are combined in this recipe compared to typical recipes? {scale: same}
5. How novel is the combination of the ingredients in this recipe compared to typical recipes? {scale: same}
6. Assuming a cook followed this recipe, would people want to taste it? {scale (1–5): 1: Never, 2: Only if they have to, 3: Only if they're really hungry, 4: They'd probably try it, 5: Yes, definitely}
7. Assuming a cook followed this recipe after the required modifications, would people want to taste it? {scale: same}
8. How original do you find this recipe overall? (after the required modifications) {scale (1–5): 1: Not original at all, 2: Slightly original, 3: Somewhat original, 4: Fairly original, 5: Extremely original and creative}