Generating Distinctive Recipe Names via Relative Feature Comparison in Recipe Set

Maoto Watanabe[1] and Yoshiyuki Shoji[1][0000−0002−7405−9270]

Shizuoka University, Hamamatsu, Shizuoka 432–8011, Japan
watanabe.maoto.21@shizuoka.ac.jp, shojiy@inf.shizuoka.ac.jp

Abstract. This paper proposes a method for generating expressive and distinctive recipe names by identifying each recipe's unique features relative to others in the same collection. For example, when most recipes boil pasta in a pot, our method may generate a descriptive recipe name like "One-Pan Carbonara" for a recipe that completes the dish using a single frying pan only. The method detects ingredients, cooking procedures, and utensils that are statistical outliers, either significantly more or less frequent compared to the rest of the recipe set. These distinguishing features are then passed to a fine-tuned large language model, which generates the final recipe name. A user study showed that the proposed method produces accurate and appealing names that effectively highlight the distinctiveness of each recipe.

Keywords: Recipe Title · Headline Generation · Large Language Model.

1 Introduction

In recent years, user-generated recipe-sharing platforms have seen a rapid increase in the number of submitted cooking recipes. For instance, over 500,000 recipes have been posted on Food.com[1], and more than 50,000 on Allrecipes[2]. With such a vast number of available recipes, it has become increasingly challenging for users to find one that suits their preferences or cooking needs efficiently.

One contributing factor is that user-submitted recipes often have names that fail to convey the actual characteristics of the dish. Many recipes are given vague or generic names, such as "Delicious Carbonara" or "Homestyle Chicken Rice," which provide little information about ingredients, cooking methods, or distinctive features. As a result, even when a recipe includes unique ingredients or preparation steps, users cannot infer those details from the recipe name alone. For example, on Food.com, there are over 300 different recipes for Carbonara, a simple cream-based pasta, making it hard to distinguish among them based solely on their names. Consequently, users are often forced to click through numerous recipes and read the full content to determine whether a recipe meets their specific needs.

[1] Food.com: https://www.food.com/
[2] allrecipes: https://www.allrecipes.com/

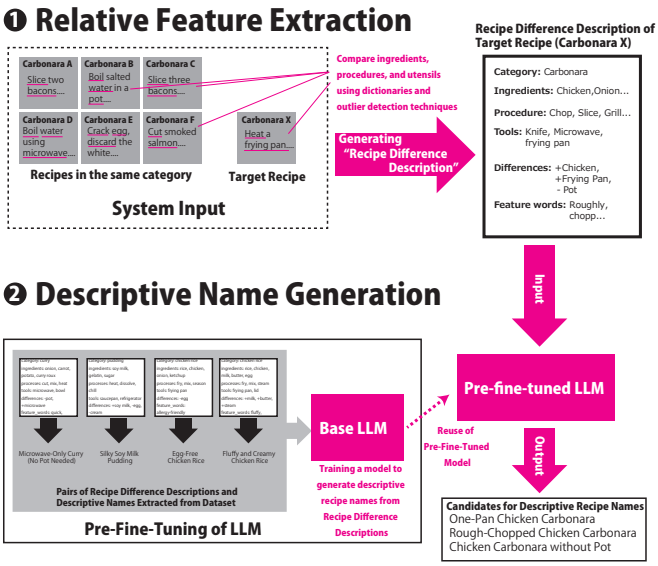

Fig. 1. An input-output example of the proposed method. Features are extracted from the recipe, a recipe-difference feature description is created, and the model generates a recipe name based on that input.

To address this search difficulty, we propose a method for generating concise recipe names that highlight each recipe's distinctive features. A key observation is that the characteristics of a recipe are best understood not in isolation, but in comparison with other recipes within the same category. For example, stating that a dish "uses eggs" may be informative when most similar recipes do not, as in the case of labeling a pasta dish "Egg-Based Peperoncino" (*i.e.*, most Peperoncino do not use egg). However, for a dish like the "Omurice", a Japanese fried rice wrapped in omelet, where eggs are a common ingredient across recipes, a name such as "Omurice with Egg" fails to convey any distinctiveness. This illustrates the importance of identifying features in a relative context, highlighting what distinguishes a recipe from others in the collection.

Even when such relative features are successfully identified, they must be expressed in a way that is both concise and understandable to human users in order to be practically useful. For instance, in the pasta category, most recipes boil pasta in a pot. However, some recipes boil the pasta directly in a frying pan and complete the dish without using a pot at all. In this case, the absence of a pot serves as the distinguishing feature, and such recipes are often referred to as "One-Pan Pasta." For users unfamiliar with recipe search or culinary terminology, articulating these features in a short and meaningful phrase can be challenging. Therefore, assigning appropriate descriptions requires not only identifying relative features but also generating natural and intuitive language to describe them.

To verbalize the relative features of a recipe, we adopt a generative large language model (LLM). Figure 1 shows an overview of our method. Our approach

takes as input a target recipe and a collection of recipes in the same category. It extracts a brief description of the ingredients, cooking procedures, and utensils that distinguish the target recipe from others. Based on these distinguishing features, it generates and ranks multiple candidate names that reflect the recipe's distinctiveness within the category. To ensure that the generated names are easy to understand and natural-sounding, we fine-tune the large-scale pre-trained language model T5 (Text-to-Text Transfer Transformer) [5] on a set of comparative descriptions and recipe names.

We performed data cleansing on a large-scale Japanese recipe dataset and fine-tuned the model using the processed data. To ensure high-quality training signals, we selected only those recipes whose names and content exhibited clear and distinctive features. This enabled the model to learn to generate recipe names that effectively reflect the unique characteristics of previously unseen recipes.

We evaluated our method through a subject experiment to assess whether the generated names accurately captured the relative distinctiveness of each recipe. Participants rated how well each generated name reflected the uniqueness of the target recipe within its comparison set. The experimental results demonstrate the effectiveness of our approach in identifying and verbalizing distinctive features in recipe naming.

2 Related Work

This study aims to extract the relative features of a given recipe and generate a natural-language description that reflects those features. To achieve this, we compare the target recipe with others in the same category and construct a "recipe difference description" in an interpretable format. This process can be regarded as a form of abstractive text summarization guided by contrastive analysis.

2.1 Text Feature Analysis

Various studies have investigated the extraction of distinctive features from text, beyond the domain of cooking recipes. Yan *et al.* [11] proposed a method that uses large language models (LLMs) to compare pairs of documents and select the more appropriate one. Similarly, Zhong *et al.* [12] developed a technique to generate feature descriptions from two document sets using LLMs. Unlike these approaches, our study utilizes LLMs solely for description generation; the extraction of recipe features is performed using an outlier-based method.

Wan *et al.* [8] proposed a technique for document classification based on identifying bi-grams that capture distinctive textual features. Setiawan *et al.* [6] improved classification performance in bullying detection by combining TF-IDF scores with word occurrence statistics and task-specific pre-filtering. In contrast, our work analyzes textual features not for classification, but to generate recipe names that reflect relative distinctiveness.

2.2 Cooking Recipe Analysis and Application

Cooking recipes differ from general text due to their domain-specific structure, which complicates the application of conventional similarity-based methods.

Wang *et al.* [9] represented recipes as workflow graphs to compute similarity. Jermsurawong *et al.* [4] proposed a tree-structured representation of recipes to support recipe retrieval. These studies transformed recipe text into structured formats to enhance interpretability. Our approach similarly transforms recipes, but focuses on clarifying distinctive features by constructing a "recipe difference description."

Our method extracts characteristic elements by comparing a single recipe against others in the same category. Hanai *et al.* [3] extracted ingredients from recipe names and identified distinctive ingredients based on their frequency within a category. Yamakata *et al.* [10] proposed a method to extract features from workflow graph representations of recipes. In contrast, our work considers the absence of ingredients or utensils as distinctive features, and translates these features into natural language recipe names.

2.3 Abstractive Summarization

Generating recipe descriptions from input recipes is closely related to abstractive summarization. Su *et al.* [7] proposed a two-stage method in which extractive summaries of text segments are generated first, followed by abstractive summarization using a language model. Dou *et al.* [2] introduced a method that incorporates extractive summaries as guiding signals for the generation process. Similarly, our approach does not directly summarize recipe text. Instead, we first convert each recipe into a "recipe difference description," which is then used as input to a language model to generate an abstractive name. Whereas previous work primarily employed automatic evaluation, our study includes human evaluation. We adopt conventional evaluation criteria such as accuracy, fluency, and attractiveness, and additionally introduce a new perspective: the degree to which the generated name reflects the recipe's distinctive features.

3 Generating Recipe Names Reflecting Relative Recipe Features

In this section, we describe the full method for generating a Descriptive Recipe Name that reflects the relative features of a target recipe within a given set of recipes. As illustrated in Figure 1, the proposed method consists of two main components:

1. generating a Recipe Difference Description (RDD), and
2. generating a recipe name using a large language model.

However, to realize this process, both preprocessing and postprocessing steps are required. Therefore, we explain the method step by step, in the manner of a cooking recipe.

First, we construct domain-specific dictionaries. These dictionaries are then used to extract the relative features of the target recipe and generate the corresponding RDD. Next, we perform data cleansing to construct high-quality training data, and fine-tune the LLM using this data. Once the model is trained, it generates multiple candidate recipe names from a given input RDD. Finally,

the candidates are ranked based on typicality within the generated set, and the top-ranked name is selected as the final Descriptive Recipe Name.

3.1 Constructing Cooking Term Dictionaries

Before extracting recipe features, we constructed domain-specific dictionaries to facilitate the effective processing of user-submitted recipes. Specifically, we created dictionaries for identifying ingredients, cooking procedures, and utensils mentioned in recipe texts. The entries were collected semi-automatically, primarily from structured metadata on recipe-sharing websites, Wikipedia, and other sources containing enumerated lists of relevant terms.

The ingredient dictionary was constructed from metadata obtained from a recipe-sharing website. Each recipe typically includes, as metadata, a structured list of required ingredients. We cleansed and normalized this metadata to construct the dictionary. Ingredients that appeared infrequently or exhibited inconsistent notation were removed during preprocessing.

The dictionaries for cooking procedures and utensils were built using Wikipedia as the primary source. We extracted cooking procedure terms, such as "shred," "dice," "boil," and "fry," from section headings in Wikipedia articles. We focused on cooking-related categories and subpages (*e.g.*, "Cooking Techniques"[3], "Japanese Cooking Utensils"[4]). The extracted lists were manually refined: only appropriate verb forms were retained for cooking procedures and were converted into standardized forms. The utensil list was constructed with fine granularity, capturing even small distinctions (*e.g.*, several variations of cooking knives).

3.2 Extracting Relative Features and Constructing Recipe Difference Descriptions (RDD)

To generate recipe names that reflect a recipe's distinctive characteristics, our method first constructs a textual representation of the recipe's relative features, called a Recipe Difference Description (RDD). Figure 2 illustrates the structure of a Recipe Difference Description (RDD). It is a plain-text format with field names followed by colons and comma-separated terms, consisting of single-recipe features extracted using dictionaries and relative features identified via outlier detection. This RDD captures how the target recipe differs from other recipes in the same category, focusing on ingredients, procedures, and utensils.

Identifying Basic Recipe Elements: We begin by extracting basic elements from each recipe: the category, ingredients, procedures, and tools. The category is obtained from metadata or inferred from the recipe name. To extract ingredients,

[3] Wikipedia "Cooking Technique" (in Japanese)
https://ja.wikipedia.org/wiki/%E8%AA%BF%E7%90%86%E6%B3%9

[4] Wikipedia "Japanese Cooking Utensils" (in Japanese)
https://ja.wikipedia.org/wiki/%E6%97%A5%E6%9C%AC%E3%81%AE%E8%AA
url%BF%E7%90%86%E5%99%A8%E5%85%B7

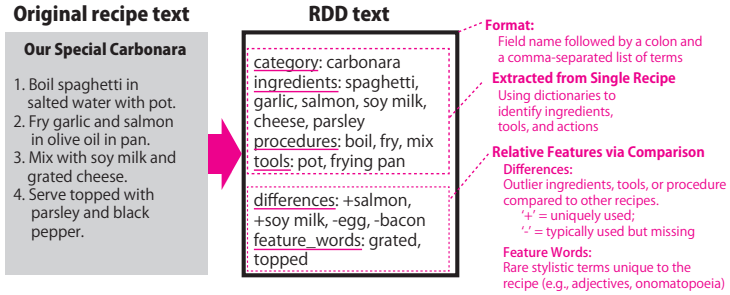

Fig. 2. Example of Recipe Difference Description (RDD). An RDD is generated from the original recipe text by extracting basic features and identifying relative differences compared to other recipes.

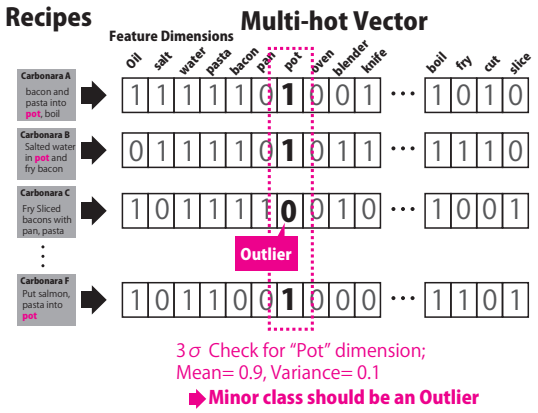

Fig. 3. Example of outlier detection with multi-hot vectors. Recipes are represented as multi-hot vectors. Outliers, such as a recipe missing "pot," are detected using the 3σ rule within each category.

procedures, and tools, we use the dictionaries constructed in the previous step. We perform morphological analysis on the recipe text and extract any tokens that match dictionary entries, classifying them accordingly.

Identifying Relative Feature using Outlier Detection Next, we identify which elements in the target recipe are relatively distinctive compared to others in the same category. We define two types of relative features:

- Inclusion-based differences: elements that are rare across the category but present in the target recipe, and
- Exclusion-based differences: elements that are common in the category but missing in the target recipe.

For example, if a "Carbonara" recipe uses chicken instead of the usual pork pancetta, or omits the use of a pot, these are treated as relative differences.

To identify what makes a recipe distinctive, we adopted the Outlier Detection technique, such as the 3σ method, as shown in Fig. 3. To enable such outlier detection, each recipe is represented as a multi-hot vector, where each dimension corresponds to an ingredient or other feature. Let R be the set of recipes in a given category, and let each recipe be denoted as $r_i \in R$. We represent each recipe r_i as a binary vector $\mathbf{v}(r_i) = (v_{i1}, v_{i2}, \ldots, v_{in})$, where each dimension $v_{ij} \in \{0, 1\}$ indicates whether the j-th element (ingredient, tool, or action) is present in the recipe.

We define the average frequency of element j across the category as:

$$\sigma(R, n) = \sqrt{\frac{\sum_{r_i \in R} (\sum \frac{\mathbf{v}_{r_{in}}}{|R|} - \mathbf{v}_{\mathbf{r_{in}}})^2}{|R|}}, \tag{1}$$

where the average value AVG of each dimension (*i.e.*, $\sum \frac{\mathbf{v}_{r_{in}}}{|R|}$), and define outliers using the following condition:

$$\mathbf{v}_{\mathbf{r_{in}}} < AVG - 3\sigma, \mathbf{v}_{\mathbf{r_{in}}} > AVG + 3\sigma. \tag{2}$$

Now that outlier-based features have been identified, they are compiled into a single RDD text. As illustrated in the figure 2, the RDD is structured as a list of field names and their corresponding values. If the outlier condition is satisfied, the corresponding ingredient, action, or utensil is marked as a difference indicator. We denote such features with plus sign if they are present in the recipe but rare in the category (*e.g.*, +pot if only the target recipe uses a pot while most others do not), and with minus sign if they are absent from the recipe but common in others (*e.g.*, -egg if the recipe is unusual in not using eggs).

In addition to these structured features, we extract feature terms, such as rare words that appear in the recipe but are uncommon within its category. Unlike difference indicators, which are limited to predefined dictionaries, feature terms are selected from the entire recipe text. To identify them, we tokenize the recipe, compute word frequencies within the category, and select those in the bottom 10%. Words such as roughly or fluffy may be included if they are rare enough. These terms help capture stylistic or perceptual nuances in the recipe.

3.3 Dataset Cleansing for Selecting Distinctive Recipe–Name Pairs

Now that any recipe can be transformed into a Recipe Difference Description (RDD), a language model can be fine-tuned to generate recipe names from these representations. However, most user-generated recipes are either unremarkable or assigned generic names that fail to capture their distinctive characteristics. For instance, even a creative variation of carbonara may be titled "Our Family's Carbonara" or "Tasty Carbonara," offering no indication of its uniqueness.

To construct effective training data, we first identified recipes with content-level distinctiveness by counting their difference indicators (as defined in Section 3.2) and computing the mean and variance per category. Recipes exceeding the threshold based on the 3σ rule were retained.

Among these, we further selected those with distinctive recipe names. For each word in a name, its frequency within the category and across all categories was computed. To emphasize category-specific uniqueness, a distinctiveness score was calculated as the ratio of category frequency to global frequency.

For example, "salmon" frequently appears in recipe names within the sushi category, but rarely in Caesar salad. A recipe named "Grilled Salmon Caesar Salad" would therefore include "salmon" as a distinctive term, illustrating the importance of category-aware scoring.

We computed the overall distinctiveness of a recipe name by summing the scores of its component words. Formally, let $C = \{C_1, C_2, \ldots, C_k\}$ be the set of categories, $R(c)$ the set of recipes in category c, and W the vocabulary. For a word $t \in W$, defined as:

$$f_{cat}(t, c) = \frac{\sum_{r \in R(c)} f_{rec}(t, r)}{\sum_{r \in R} f_{rec}(t, r)}. \tag{3}$$

We used this score to select recipe–name pairs with high recipe name distinctiveness as training data.

3.4 LLM Fine-tuning and Recipe Name Candidate Generation

Given the large set of recipe pairs, each consisting of a Recipe Difference Description (RDD) and a corresponding distinctive recipe name, we fine-tune a language model to generate recipe names that reflect relative characteristics.

We employ T5 (Text-to-Text Transfer Transformer), a large language model specifically designed for text-to-text transformation tasks. Since T5 naturally supports input-output text mapping, the model can be trained in the same format as used during inference. Specifically, the RDD is used as input, and the distinctive recipe name as the target output.

The fine-tuned model learns to generate recipe names that express salient and distinctive traits, even for previously unseen recipes. To obtain diverse candidate names for a given recipe, the model can generate multiple outputs by varying the random seed. In practice, a target recipe is first converted into its RDD form, which is then passed to the model to produce a list of name candidates.

Examples of generated recipe names are shown in Table 1. These include expressions that articulate unique aspects of the recipe, such as indicating the use of leftovers or frozen rice, whereas the original recipe names merely included generic phrases like "easy" or "home-style."

3.5 Typicality-Aware Re-ranking of Generated Recipe Names

The first candidate generated by a language model is not always the most appropriate. To improve the quality of recipe names, we generate multiple candidates and re-rank them to select the most representative one. The re-ranking is based on two criteria: whether the candidate includes the category name (*e.g.*, "carbonara"), and how typical the candidate is among all generated recipe names.

Table 1. Examples of recipe names generated from RDDs using the fine-tuned model.

Original name (translated from Japanese)	Generated name (translated from Japanese)
Pork Belly and Tomato Negi-Salt Peperoncino	Pork Belly and Tomato Peperoncino
First-Ever Stuffed Cabbage Rolls	Pressure-Cooked Stuffed Cabbage Rolls
Easy and Delicious Braised Pork	Braised Pork in a Pressure Cooker
Nostalgic Hayashi Rice(^^)	Flavorful Meat♪Hayashi Rice
Salmon Meunière with Lemon Soy Sauce	Salmon Meunière with Mushroom Butter Soy Sauce
Easy Chicken&Tomato Risotto	Spicy Chicken and Onion Risotto
Crushed Coffee Jelly	Coffee Jelly with Milk Syrup Using Agar Powder
Simple Meat and Potato Pouch with Leftovers	Easy Salad with Leftover Meat and Potatoes
Easy Curry Risotto	Tomato Risotto with Frozen Rice

Specifically, we generate 100 candidate recipe names for each input. Candidates that do not include the category name are first filtered out. For example, if the input belongs to the "carbonara" category, we discard any name that does not contain the word "carbonara."

Then, we calculate a typicality score for each remaining recipe name based on the frequency of words within the candidate set. Let $T = t_1, t_2, ..., t_n$ be the set of generated names, and $w \in W$ a word in the names. The typicality score $s(t_i)$ of a name t_i is defined as:

$$s(t_i) = \sum_{w \in t_i} f(w) \tag{4}$$

where $f(w)$ denotes the frequency of word w in T. The name with the highest score is selected as the final output. This enables the system to reliably generate recipe names that highlight how the target recipe differs from others in the set, while ensuring that the final output includes the appropriate category name through a typicality-based re-ranking process.

4 Evaluation

To evaluate the effectiveness of the proposed recipe name generation method, we conducted both automatic and human evaluations. The automatic evaluation focuses on the linguistic quality of the generated names. In contrast, the human evaluation examines whether the names are appropriate and appealing as recipe names. In particular, it evaluates how well each name reflects the distinctive characteristics of the corresponding recipe within its category. To support these evaluations, we prepared a large-scale recipe dataset and implemented a complete system capable of generating a recipe name for any given input.

4.1 Dataset

We used the Cookpad Dataset, provided by Cookpad Inc. and the National Institute of Informatics [1], which contains over 1.7 million user-submitted Japanese recipes, including names, ingredient lists, and preparation steps. To assign a recipe category to each entry, we compiled a list of 261 general dish names, such as "beef stew" and "fried rice", and assigned recipes to a category if their names

included any of these terms. Recipes that could not be categorized in this way were excluded from further processing.

For training, we selected a subset of 2,406 recipes that exhibited distinctive characteristics both in their content and their recipe name, based on an outlier-based filtering approach (see subsection 3.3). These were used to fine-tune our recipe name generation model.

4.2 Baseline Methods

To clarify the contribution of each component in our proposed method, we conducted an ablation study using several variant models. Each variant removes one or more specific components from the full proposed pipeline, allowing us to assess which design choices are most effective and how they affect the overall performance.

The baseline methods are as follows:

- **Without Differences**: A variant that removes the relative difference indicators (*e.g.*, +microwave, -pot) from the input.
- **Without Feature Words**: A variant that omits stylistic or perceptual feature terms (*e.g.*, "fluffy", "roughly") from the input.
- **Full Recipe Text**: A variant that uses the entire recipe text as input, without any structured representation such as recipe difference descriptions.
- **Random Training Data**: A variant trained on 2,406 recipes randomly sampled from the dataset, instead of selecting ones with distinctive content and names.

By comparing the output quality and evaluation scores of these variants against the proposed method, we aim to reveal the role each design component plays, such as the value of contrastive features, stylistic cues, and curated training data, in generating accurate and expressive recipe names.

4.3 Implementation

To evaluate the proposed method, we implemented a system that trains the model and generates recipe descriptions. We used the Hugging Face Transformers library[5], which provides support for transformer-based models, to implement T5. We used a Japanese pretrained version of T5[6] as the base model.

For fine-tuning, we adopted the default parameter settings provided by Hugging Face Transformers. The maximum number of tokens for the input was set to 512, and for the output to 64. Fine-tuning was conducted using the dataset described in Section 4.1.

Since the dataset consists of Japanese texts, word boundaries are not explicitly marked by whitespace as in English. Therefore, we applied morphological

[5] Hugging Face "Hugging Face Transformers" https://huggingface.co/docs/transformers/index

[6] Hugging Face "sonoisa/t5-base-japanese" https://huggingface.co/sonoisa/t5-base-japanese

Table 2. ROUGE scores for automatically generated recipe names

Method	ROUGE-1	ROUGE-2	ROUGE-L
Proposed Method	0.40	0.18	0.38
Without Differences	0.41	0.18	0.39
Without Feature Words	**0.55**	**0.38**	**0.54**
Full Recipe Input	0.40	0.15	0.38
Random Training Data	0.41	0.18	0.39

analysis to segment recipe texts and names into word units. We used MeCab[7], a widely-used Japanese morphological analyzer, for this task. To enhance coverage of contemporary and domain-specific terms, we used the mecab-ipadic-NEologd[8] dictionary, which incorporates many neologisms and frequently updated vocabulary.

4.4 Automatic Evaluation of Recipe Name Linguistic Quality

We conducted an automatic evaluation to assess the linguistic quality of the generated recipe names. While our primary goal is to generate names that reflect the distinctive features of each recipe, it is also important to examine whether the outputs are fluent and reasonably consistent with names written by actual users.

To this end, we used ROUGE scores, a standard metric in abstractive summarization, to measure the textual similarity between generated names and user-assigned names in the dataset. It is worth noting, however, that user-generated recipe names do not always reflect the relative uniqueness of each recipe. Therefore, ROUGE scores provide only a surface-level approximation of output quality, rather than a true measure of distinctive feature reflection.

The calculated ROUGE scores are shown in Table 2. Among all methods, the variant without feature words achieved the highest scores across all ROUGE metrics, likely due to its tendency to produce concise names closely aligned with those commonly found in the dataset.

4.5 Human Evaluation of Recipe Name Appropriateness

To assess whether the generated recipe names are appropriate as recipe names, we conducted a human evaluation focusing on three key aspects: fluency, attractiveness, and the degree to which the name reflects the distinctive features of the recipe. Unlike automatic metrics, which measure textual similarity to reference names, human judgment is necessary to evaluate whether a name truly highlights what makes a recipe unique within its category. This is particularly important in our task setting, where user-assigned recipe names in the dataset often fail to express relative distinctiveness, making automatic evaluation insufficient for a complete assessment.

[7] MeCab
https://taku910.github.io/mecab/

[8] mecab-ipadic-NEologd
https://github.com/neologd/mecab-ipadic-neologd

To evaluate whether each generated recipe name successfully conveys what makes the target recipe distinctive, we designed a comparative evaluation setting. The key idea was to allow participants to assess the name's appropriateness from multiple perspectives, including whether it accurately reflects the recipe's ingredients, reads fluently, appears attractive, and captures distinctive features in comparison to similar recipes.

In each trial, participants were shown one target recipe and five comparison recipes randomly selected from the same category. After reviewing the recipes, they were presented with a generated recipe name and asked to evaluate it based on the four criteria above. The experiment was conducted using ten different target recipes, each evaluated using recipe names generated by the proposed method and four baseline methods.

To comprehensively assess the quality and appropriateness of each generated recipe name, we designed four evaluation criteria. These criteria were chosen to cover both surface-level and semantic aspects of recipe names, including lexical correctness, fluency, persuasive appeal, and the ability to express relative distinctiveness within a recipe category.

Two participants rated each generated recipe name using the following four criteria:

1. **Ingredient Accuracy**: Are the ingredients mentioned in the name actually used in the recipe?
2. **Fluency**: Is the name natural and grammatically correct in Japanese?
3. **Attractiveness**: Does the name convey the appeal of the recipe?
4. **Feature Reflection**: Compared to other recipes, does the name highlight what makes this recipe distinctive?

Each question was rated on a 5-point scale. For ingredient accuracy, we adopted a flexible interpretation to account for synonyms and ingredient granularity. For example, if a recipe uses "king crab" but the name simply says "crab," the rating may still reflect partial correctness. Conversely, if a recipe only uses standard crab meat, but the name claims "king crab," the rating would be reduced accordingly. The 5-point scale allows for such nuanced evaluation. Given the substantial inter-rater reliability, Cohen's$\kappa = 0.65 > 0.60$, two raters were deemed sufficient.

For fluency and attractiveness, participants evaluated whether the name was natural in Japanese and whether it expressed some form of appeal beyond generic terms. For instance, while phrases like "super tasty" may sound appealing, they are too vague to convey meaningful information. A good name should highlight a specific quality of the dish, such as "crispy," "spicy," or "oven-baked."

For feature reflection, participants first identified what distinguished the target recipe from the five comparison recipes, then judged whether the recipe name captured that difference. For example, in the "lasagna" category, if most comparison recipes use beef but the target recipe uses eggplant and no meat, this vegetarian variation would be considered distinctive. Participants then evaluated whether the recipe name reflected this uniqueness, such as by including "vegetarian" or "eggplant."

Table 3 shows the average scores for each method across the four evaluation criteria. The proposed method achieved the highest scores in three out of

Table 3. Average scores for each evaluation criterion (1–5 scale). Asterisks indicate statistically significant differences from the proposed method (* $p < 0.05$, ** $p < 0.01$) in Student's t-test.

Evaluation Criterion	Ingredient Accuracy	Fluency	Attractiveness	Feature Reflection
Proposed Method	**4.58**	3.60	**3.60**	**3.83**
Without Differences	**3.78	**3.78**	*3.03	**2.70
Without Feature Words	*4.10	3.20	3.08	**3.00
Full Recipe Input	**3.23	3.95	**2.28	**2.28
Random Training Data	4.35	3.45	3.28	*3.33

four criteria: Ingredient Accuracy (4.58), Attractiveness (3.60), and Feature Reflection (3.83). In contrast, methods that removed relative difference indicators or stylistic feature words showed significant drops, particularly in Feature Reflection. The variant using full recipe text performed worst in most categories, despite having the longest input.

These results suggest that both the structured input format and the curated training data contributed to the improved quality of the generated recipe names, especially in capturing what makes each recipe distinctive.

5 Discussion

In this section, we discuss the results obtained from the evaluation experiments. In the ROUGE score evaluation, the method that excluded feature words from the input achieved the highest score. Recipe names generated using the method without feature words tended to be short and concise overall. Specifically, many of the recipe names consisted of a combination of ingredient names and dish names, such as "Mackerel Peperoncino." Since the actual recipe names given to most recipes typically include both ingredient and dish names, this likely led to higher ROUGE scores.

However, even when recipe names express similar characteristics, ROUGE is unable to properly evaluate them if they are described differently. For example, recipes that do not use a frying pan or pot might have recipe names containing expressions like "quick", "easy", or "no heat required". Although these all reflect the same feature, ROUGE would treat them as mismatches because the exact words differ. On the other hand, concise recipe names composed of ingredients and dish names tend to result in higher ROUGE scores. When less input information is provided, it is more likely that the ingredient names used in the recipe are directly reflected in the generated recipe name.

Next, we discuss the results obtained from the human evaluation. For ingredient accuracy, the recipe names generated by the method that directly inputs the entire recipe scored the lowest. This suggests that simply feeding the full recipe into the language model does not result in accurate reflection of ingredient names. Although this approach had the longest input length compared to other methods, the proposed method, which uses a more structured input, out-

performed even the shorter "without differences" method in terms of ingredient accuracy, demonstrating the effectiveness of our approach.

We now turn to the evaluation of attractiveness and feature reflection. The proposed method achieved the highest scores for both criteria, while the recipe-based method had the lowest. This suggests that better feature reflection may contribute to higher perceived attractiveness. For instance, a recipe name like "One-Pan Pasta" highlights the feature of requiring only a frying pan, which can make the recipe more appealing. Thus, improving feature reflection likely enhances attractiveness as well.

Titles that scored highly for both attractiveness and feature reflection often included ingredient names that are uncommon in that category. For example, in a hamburger recipe using shiso leaves and zucchini, the recipe name that accurately reflected both the ingredients and the dish name received a high evaluation. Moreover, not only unique ingredients but also the absence of certain cooking tools or processes contributed to perceived simplicity and higher scores. In the case of a fried rice recipe using frozen rice, recipe names containing terms like "easy" or "time-saving" were rated highly in both attractiveness and feature reflection. This recipe did not involve cooking rice, so the absence of elements like "rice cooker" or "cooking rice" appeared in the difference expressions and enabled the model to verbalize that feature in the generated recipe name.

Overall, the results confirm that incorporating relative difference indicators and stylistic features into the input leads to more informative and appealing recipe names. These findings highlight the importance of explicitly modeling contrastive features when generating names that are both expressive and contextually appropriate.

6 Conclusion and Future Work

This paper presented a method for generating expressive and distinctive recipe names by identifying what sets a given recipe apart from others in the same category. We introduced the notion of a Recipe Difference Description (RDD), a structured textual representation that captures relative differences in ingredients, procedures, and utensils. By fine-tuning a language model on carefully curated training data, we enabled the generation of recipe names that are not only fluent and appealing but also reflective of the recipe's unique characteristics.

Through both automatic and human evaluation, we demonstrated that explicitly modeling contrastive features leads to improved accuracy and expressiveness in generated names. In particular, user studies confirmed that recipe names generated by the proposed method were more successful in capturing the distinguishing features of each dish.

Although we focused on cooking recipes, the proposed approach, verbalizing an item's relative distinctiveness within a set, has potential applications in other domains. Examples include naming products, summarizing research papers, or highlighting unique characteristics in content recommendation systems. Future work will explore these extensions and investigate how contrastive summariza-

tion techniques can support more general forms of personalized or context-aware generation.

Acknowledgements

This work was partially supported by JSPS KAKENHI under Grant Numbers 25K03229, 25K03228, and 24K03228. We used "Cookpad dataset" provided by Cookpad Inc. via IDR Dataset Service of National Institute of Informatics [1].

References

1. Cookpad Inc.: Cookpad data. https://doi.org/10.32130/idr.5.1 (2015), informatics Research Data Repository, National Institute of Informatics. (dataset)
2. Dou, Z.Y., Liu, P., Hayashi, H., Jiang, Z., Neubig, G.: Gsum: A general framework for guided neural abstractive summarization. In: Proc. of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 4830–4842 (2021)
3. Hanai, S., Nanba, H., Nadamoto, A.: Clustering for closely similar recipes to extract spam recipes in user-generated recipe sites. In: Proc. of the 17th International Conference on Information Integration and Web-based Applications & Services. pp. 1–5 (2015)
4. Jermsurawong, J., Habash, N.: Predicting the structure of cooking recipes. In: Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 781–786 (2015)
5. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research **21**(140), 1–67 (2020)
6. Setiawan, Y., Gunawan, D., Efendi, R.: Feature extraction tf-idf to perform cyberbullying text classification: A literature review and future research direction. In: Proc. of 2022 International Conference on Information Technology Systems and Innovation (ICITSI). pp. 283–288. IEEE (2022)
7. Su, M.H., Wu, C.H., Cheng, H.T.: A two-stage transformer-based approach for variable-length abstractive summarization. IEEE/ACM Transactions on Audio, Speech, and Language Processing **28**, 2061–2072 (2020)
8. Wan, C., Wang, Y., Liu, Y., Ji, J., Feng, G.: Composite feature extraction and selection for text classification. IEEE Access **7**, 35208–35219 (2019)
9. Wang, L., Li, Q., Li, N., Dong, G., Yang, Y.: Substructure similarity measurement in chinese recipes. In: Proc. of the 17th international conference on World Wide Web. pp. 979–988 (2008)
10. Yamakata, Y., Imahori, S., Sugiyama, Y., Mori, S., Tanaka, K.: Feature extraction and summarization of recipes using flow graph. In: Social Informatics: 5th International Conference, SocInfo 2013, Kyoto, Japan, November 25-27, 2013, Proceedings 5. pp. 241–254. Springer (2013)
11. Yan, J.N., Liu, T., Chiu, J., Shen, J., Qin, Z., Yu, Y., Lakshmanan, C., Kurzion, Y., Rush, A.M., Liu, J., et al.: Predicting text preference via structured comparative reasoning. In: Proc. of the 62nd Annual Meeting of the Association for Computational Linguistics. pp. 10040–10060 (2024)
12. Zhong, R., Snell, C.B., Klein, D., Steinhardt, J.: Describing differences between text distributions with natural language. In: Proc. of the International Conference on Machine Learning (2022)