University of Cyprus

DSC 514: Natural Language Processing

Smart Chef: Recipe Creation Using Fine-Tuned Language Models

Panagiotis Michael Fivos Theodoridis Maysam Khatib
Rafail Athos Panagi Constantinos Odysseos

May 8, 2025

1 Introduction

In a world where artificial intelligence is transforming everyday lives, the food industry is no exception. The ability to create recipes dynamically based on available ingredients can be an extremely useful tool for both unprofessional cooks and professional chefs. This study investigates the use of large language models (LLMs) for automated recipe development, with a focus on fine-tuning these models to recognize and respond to ingredient-based recommendations.

To achieve this purpose, the project includes several natural language processing (NLP) parts such as preprocessing, recipe categorization, nutritional estimate, and data-driven analysis of existing recipe patterns. Recipes are enhanced with information such as predicted caloric levels and organized into meaningful food groupings to enable more personalized and informative generation. Exploratory data analysis is used to identify relevant trends and insights in a dataset, which guide preprocessing and modeling decisions.

The final stage is fine-tuning the large language model to create new recipes depending on certain elements. The system's goal is to develop coherent, varied, and practical recipes by integrating structured metadata with generative capabilities. This work not only proves the viability of LLMs in food applications, but it also emphasizes the importance of incorporating structured food information into generative NLP models.

2 Data Overview

The **RecipeNLG** dataset, a comprehensive, structured collection of 2,231,142 English-language cooking recipes, was used in this project [1]. Every recipe entry in the collection has several fields, such as:

- **title**: Recipe's name.
- **ingredients**: List of ingredients in string format, which usually includes quantities and measurements.
- **directions**: Set of culinary instructions, typically organized as a list of steps.
- **link**: Recipe's source URL, which refers to the original web page.
- **source**: The website or platform on which the recipe was gathered
- **NER**: List of named entities extracted from the ingredients of the recipe, providing a more standardized perspective of key terms of food items.

Table 1 shows the first five entries of the recipe dataset, including titles, ingredient lists, cooking directions, source links, and extracted named entities.

3 Data Preprocessing

3.1 Data Cleaning

The original dataset had **2,231,141** recipe entries, each with ingredients, directions, named entities (NER), and metadata. To preserve only the key textual features, we first deleted unnecessary columns (title, link, and source).

Next, we cleaned the data by removing records with missing values or duplicates, resulting in 2,230,557 valid entries - 584 rows were deleted in total.

We then extracted three basic structural features:

- `ingredients_count`: number of listed ingredients.
- `NER_count`: number of recognized named entities.
- `directions_count`: number of cooking directions.

Entries containing zeros in any of these fields were removed to ensure data consistency.

3.2 Text Normalization

To prepare the data for modeling and nutritional analysis, we developed separate normalization stages on both the **ingredients** and **directions** features.

Ingredients Normalization

Ingredient lists frequently use inconsistent measurements, fractional quantities, and informal acronyms for short. This is how we addressed it:

- Replace Unicode fractions (e.g., $\frac{1}{2} \rightarrow 1/2$).
- Standardize units with regex patterns and a domain-specific mapping dictionary (e.g., tbsp. $\rightarrow$ tablespoon, oz $\rightarrow$ ounce).
- Clean tokens to get rid of trailing punctuation and normalize synonyms.

Table 1: Sample entries from the recipe dataset

Title	Ingredients	Directions	Link	Source
No-Bake Nut Cookies	[1 c. brown sugar, 1/2 c. evaporated milk, ...]	[In a heavy 2-quart saucepan, mix brown sugar..., ...]	cookbooks.com/Recipe-Details.aspx?id=44874	Gathered
Jewell Ball's Chicken	[1 small jar chipped beef, 4 boned chicken breasts, ...]	[Place chipped beef on bottom of baking dish..., ...]	cookbooks.com/Recipe-Details.aspx?id=699419	Gathered
Creamy Corn	[2 pkg. frozen corn, 1 pkg. cream cheese, ...]	[In a slow cooker, combine all ingredients..., ...]	cookbooks.com/Recipe-Details.aspx?id=10570	Gathered
Chicken Funny	[1 large whole chicken, 2 cans gravy, ...]	[Boil and debone chicken., Put pieces in baking dish..., ...]	cookbooks.com/Recipe-Details.aspx?id=897570	Gathered
Reeses Cups (Candy)	[1 c. peanut butter, 3/4 c. graham cracker crumbs, ...]	[Combine ingredients and press in pan..., ...]	cookbooks.com/Recipe-Details.aspx?id=659239	Gathered

- Convert mixed fractions (e.g., 1 1/2 = 1.5).
- Remove unnecessary characters such as brackets, quotation marks, and redundant spacing.

Directions Normalization

To improve consistency in model input, directions for cooking were normalized as well using:

- Lowercasing all text.
- Eliminating non-ASCII characters, emojis and HTML tags.
- Standardising temperature references (e.g., 350f = 350 degrees Fahrenheit).
- Remove trailing punctuation and excessive whitespace.
- Transforming list-style directions into continuous natural language text

These steps guaranteed that the textual material was clean, semantically consistent, and suitable for feature extraction, LLM prompting, or future modeling.

3.3 Sampling and Filtering

To manage computing resources and ensure effective processing afterwards, we picked a random subset of the cleaned dataset. To assure reproducibility, we picked a sample of **120,000** recipes using a fixed random seed (`random_state=42`).

Before sampling, any entries with zero values in any of the structural features—`ingredients_count`, `NER_count`, or `directions_count`—were removed. These entries usually indicated incomplete or faulty records.

After sampling, we removed extreme outliers in the `calories` feature. We used the inter-quartile range (IQR) approach to obtain the lower and upper bounds:

$$
\begin{aligned}
Q_1 &= \text{25th percentile of calories} \\
Q_3 &= \text{75th percentile of calories} \\
\text{IQR} &= Q_3 - Q_1 \\
\text{Bounds} &= [Q_1 - 1.5 \times \text{IQR},\ Q_3 + 1.5 \times \text{IQR}]
\end{aligned}
$$

Recipes falling outside of this range were considered nutritional outliers and were removed.

3.4 Nutritional Estimation (LLM-based Feature Engineering)

To add nutritional information to the dataset, we employed a large language model (LLM) to predict six nutritious features based on the normalized_ingredients feature.

These six features are:

- Calories (kcal)
- Protein (g)
- Carbohydrates (g)
- Fat (g)
- Fiber (g)
- Sugar (g)

We processed batch ingredient lists using the **Gemini 2.0 Flash model** via API. A well-constructed prompt told the model to follow a preset schema and output nutrient estimations in plain-text format, one line per recipe. We put in place the following to guarantee effectiveness and prevent rate limits:

- Batch processing with a configurable size (`batch_size = 5`)
- Multithreading using Python's `ThreadPoolExecutor`
- A short delay between requests to stay below the API's throughput threshold

Regular expressions were used to parse the model responses, which were then assembled into a structured DataFrame. To assure quality, extreme values in the calories field were filtered using the Interquartile Range (IQR) method, leaving only entries within an acceptable range.

These estimations were then added to the entire dataset, producing a comprehensive representation of each recipe that included both linguistic and nutritional information.

3.5 Final Feature Set Construction

After combining the normalized text fields and estimated nutritional values, we created a unified dataset ready for analysis and modelling. The resulting dataset included **114,631 recipes**, each represented by the following 14 features:

- Raw Text Fields: `ingredients`, `directions`, `NER`
- Structural Features: `ingredients_count`, `NER_count`, `directions_count`
- Cleaned Text: `normalized_ingredients`, `normalized_directions`
- Nutritional Estimates: `calories`, `protein_g`, `carbs_g`, `fat_g`, `fiber_g`, `sugar_g`

Table 2 shows the new structure and the first 5 entries on the dataset.

This enhanced and preprocessed dataset served as the starting point for many later stages of the project, such as EDA on Nutritional Features, and language model fine-tuning.

4 Exploratory Data Analysis (EDA)

4.1 EDA on the Initial Dataset

Prior to preprocessing, an exploratory data analysis (EDA) was performed to better understand the dataset's structure, variability, and possible weaknesses. While other filtering and cleaning processes were added later in the pipeline, we believe it would be beneficial to provide significant observations and visualizations from this initial investigation. These helped in decisions such as filtering by instruction length, ingredient count criteria, and detecting abnormal entries.

The original dataset has 2,231,142 rows with seven columns: `title`, `ingredients`, `directions`, `link`, `source`, and `NER`. Basic inspection revealed that all fields are of the string type.

Directions Count Analysis

We calculated the amount of directions in each recipe. The distribution is presented in Figure 1. While most recipes are brief, a few outliers have extremely long directions, some reaching more than 2,500. For example, "Moon Cakes" recipe has 2649 directions, and "Mike's NY Cheesecake 101" has 2605. The need for the filtering was confirmed by the identification of these long-directions.

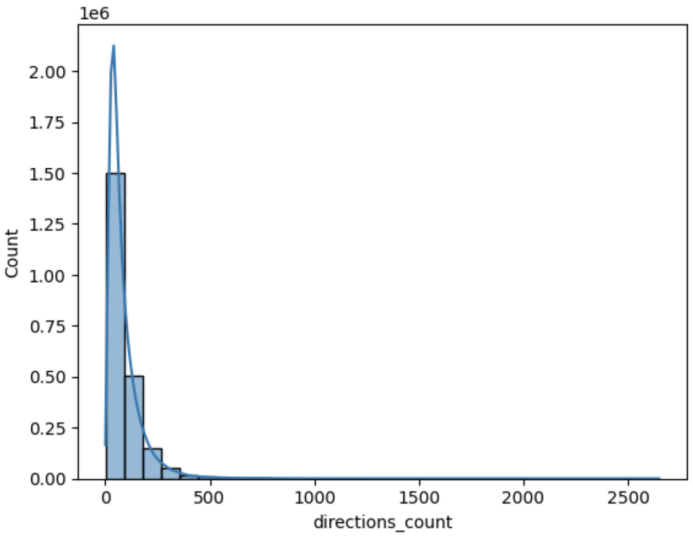

Figure 1: Distribution of Direction Word Counts

Ingredient Count Distribution

The average amount of ingredients per recipe was around 10.6, with a standard deviation of 5.59. Most recipes contained 5 to 16 ingredients, with many outliers having more than 20 ingredients reaching up to around 500. Figure 3 displays the boxplot of the ingredients count.

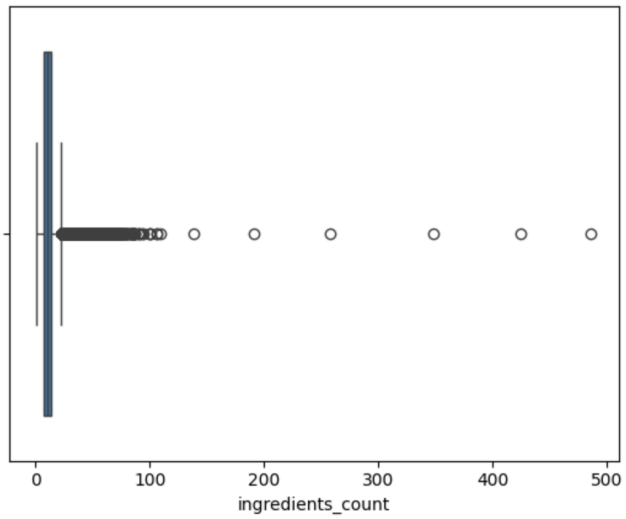

Figure 2: Distribution of Ingredient Counts

Figure 3 also shows the distribution along with standard deviation boundaries.

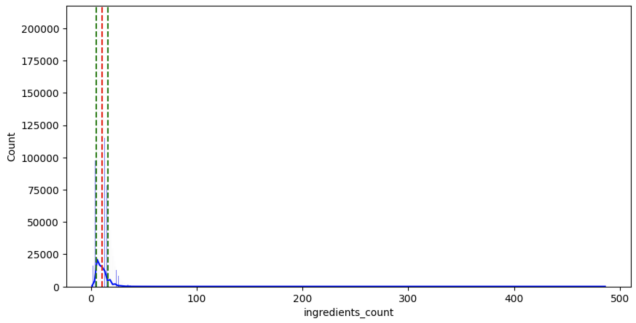

Figure 3: Distribution of Ingredient Counts with Mean and Standard Deviations

Ingredient Frequency Analysis

By aggregating all NER-identified elements, we created a frequency distribution of the most used ingredients. Figure 4 shows the top 20 most often used ingredients. It shows that basic staple like *salt*, *sugar*, *butter*, and *garlic* are the most commonly used ingredients in the dataset. These components featured hundreds of thousands of times, demonstrating their fundamental relevance in a wide range of recipe types. Some irregularities, such as trailing brackets or duplicates (e.g., `"salt"]`, `["butter"`), indicate minor tokenisation noise that have been cleaned in the preprocessing.

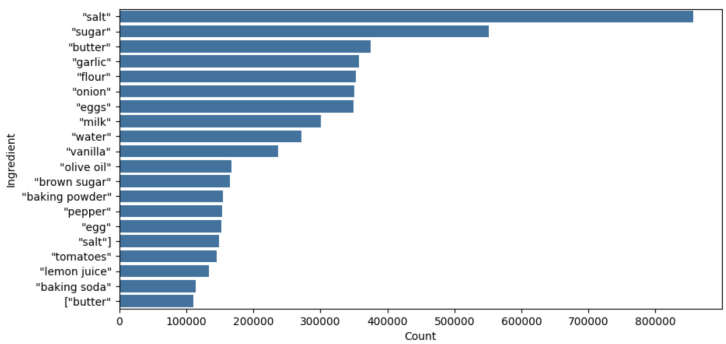

Figure 4: Most Frequent Ingredients Across the Dataset

Table 2: First 5 rows of the final preprocessed and enriched recipe dataset

ingredients	directions	NER	ingredients_count	NER_count	directions_count	normalized_ingredients	normalized_directions	calories	protein_g	carbs_g	fat_g	fiber_g	sugar_g
["6 cups rice krispies", "2 cups reese's peanu..."]	["combine the rice krispies and chopped peanut..."]	["rice krispies", "miniatures", "marshmallows"...]	8	6	219.0	6 cup rice krispies, 2 cup reeses peanut butte...	combine the rice krispies and chopped peanut b...	5950.0	85.0	750.0	300.0	25.0	480.0
["4 boneless skinless chicken breasts", "6 spr..."]	["heat the oven to 350 degree f/180 degrees c...."]	["chicken breasts", "oregano", "onion", "garli..."]	18	13	131.0	4 boneless skinless chicken breasts, 6 sprigs ...	heat the oven to 350 degree f/180 degrees cels...	2600.0	280.0	280.0	60.0	40.0	50.0
["1 (7 1/2 ounce) can refrigerated biscuits", ...]	["melt oleo.", "dip each biscuit in oleo, then..."]	["refrigerated biscuits", "oleo", "brown sugar..."]	5	5	44.0	1 (7.5 ounce) can refrigerated biscuits, 1 cup...	melt oleo, dip each biscuit in oleo, then brow...	2800.0	25.0	300.0	180.0	15.0	200.0
["2 pounds ground beef or turkey", "2 pounds r..."]	["brown meat in a kettle with a little oil", "..."]	["ground beef", "beef", "garlic", "paprika", "..."]	21	15	73.0	2 pounds ground beef or turkey, 2 pounds round...	brown meat in a kettle with a little oil, add ...	5700.0	500.0	400.0	250.0	100.0	100.0
["3 lbs beef round steak", "all-purpose flour"...]	["cut steaks into strips measuring 3 to 4 inch..."]	["beef round steak", "flour", "salt", "onion",..."]	9	8	290.0	3 lbs beef round steak, all-purpose flour, sal...	cut steaks into strips measuring 3 to 4 inches...	3500.0	400.0	80.0	150.0	10.0	30.0

Word Cloud of Ingredients

A word cloud was created from the entire ingredient list to provide a qualitative assessment of ingredients diversity. (Figure 5).

Figure 5: Word Cloud of Ingredients

4.2 EDA on Nutritional Features

Pairwise Nutrient Correlations

We first calculated pairwise Pearson correlation coefficients between all nutrient fields in order to comprehend the relationships between various nutritional indicators throughout the dataset. Figure 6 demonstrates that `calories` has a modest correlation with both `carbs` (0.75) and `sugar` (0.56), and a high positive correlation with `fat` (0.89). On the other hand, there is a modest negative association between `sugar` and `protein`, and comparatively little correlation with the majority of other measures. These patterns support predicted dietary correlations, such as the fact that sugar and fat are the main sources of calories.

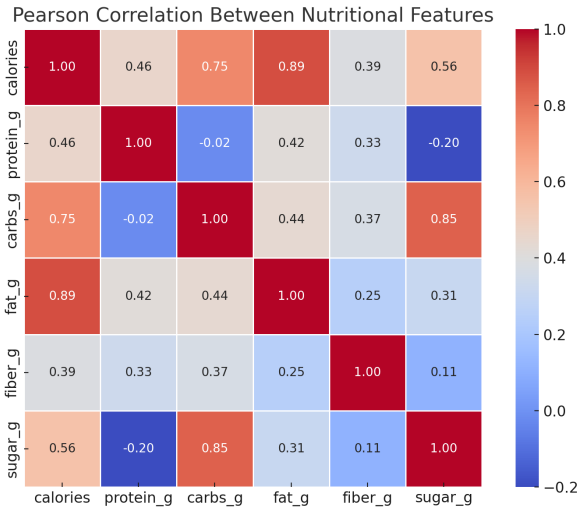

Figure 6: Pearson correlation matrix between nutritional features.

Ingredient-Level Nutrient Correlations

We next estimated correlations between particular ingredients and each nutrient value in order to conduct a more detailed analysis. This was done by aggregating quantities per ingredient across recipes and calculating correlation coefficients against each nutrient target.

This method found strong positive correlations between the caloric, carbohydrate, and sugar values of popular baking ingredients—such as `sugar`, `vanilla`, `baking soda`, and `egg`. For example, `egg` had a strong connection with both `calories` (0.30) and `fat` (0.25). Ingredients like `garlic`, `onion`, and `olive oil` showed stronger connections with `protein` and `fiber`, but often negative or neutral associations with sugar-related metrics. Table 3 shows the top 5 positively and negatively correlated ingredients by nutrient.

Table 3: Top 5 Positively and Negatively Correlated Ingredients by Nutrient

Nutrient	Top 5 Positive	Top 5 Negative
Calories	egg (0.295), vanilla (0.231), baking soda (0.218), sugar (0.196), flour (0.191)	olive oil (-0.034), lemon juice (-0.033), soy sauce (-0.032), cilantro (-0.030), red onion (-0.027)
Protein	garlic (0.242), ground beef (0.205), onion (0.175), pepper (0.156), chili powder (0.137)	sugar (-0.136), vanilla (-0.120), baking powder (-0.079), baking soda (-0.079), cinnamon (-0.070)
Carbs	sugar (0.311), baking soda (0.300), vanilla (0.297), egg (0.290), baking powder (0.263)	garlic (-0.126), olive oil (-0.097), mayonnaise (-0.092), pepper (-0.092), salt pepper (-0.078)
Fat	egg (0.252), vanilla (0.185), pecan (0.150), butter (0.143), baking soda (0.142)	cilantro (-0.033), soy sauce (-0.030), olive oil (-0.022), water (-0.019), tomatoe (-0.019)
Fiber	garlic (0.203), chili powder (0.163), onion (0.160), salt (0.158), olive oil (0.134)	vanilla (-0.012), soy sauce (-0.011), mayonnaise (-0.011), cornstarch (-0.007), milk (0.001)
Sugar	sugar (0.338), vanilla (0.330), baking soda (0.254), egg (0.224), cinnamon (0.201)	garlic (-0.184), olive oil (-0.143), pepper (-0.127), onion (-0.119), salt pepper (-0.110)

5 Food Categorization

5.1 Title-Based Clustering

We used two main grouping techniques on the recipe titles—TF-IDF vectorization and BERT embeddings—to put the recipes into groups based on topics. These techniques enabled us to identify natural groupings such as desserts, salads, casseroles, and pasta-based foods based solely on title content.

5.1.1 TF-IDF Vectorization with K-Means

The pre-processed recipe titles were converted into numerical vectors using TF-IDF with a vocabulary size of 1000 and English stop words removed. These vectors were then grouped using the K-Means technique, with $k = 10$ groups. Figure 7 depicts word clouds displaying the most prominent terms in each cluster. Clear theme patterns appeared, such as clusters dominated by "salad", "casserole" or "pudding".

Figure 7: TF-IDF based K-Means Clustering Word Clouds (10 Clusters)

5.1.2 BERT Embeddings with K-Means

To capture deeper semantic links, we used the 'all-MiniLM-L6-v2' BERT model to build embeddings for a sample of 25,000 titles. These embeddings were clustered using K-Means with $k = 20$, as calculated by the elbow method (Figure 8). The resulting clusters, visualized as word clouds (Figure 9), captured broader and more diverse categories such as cakes, sauces, meat dishes, and breakfast items. The silhouette score of **0.0554** indicates modest cohesion given the dataset's high variability and complexity.

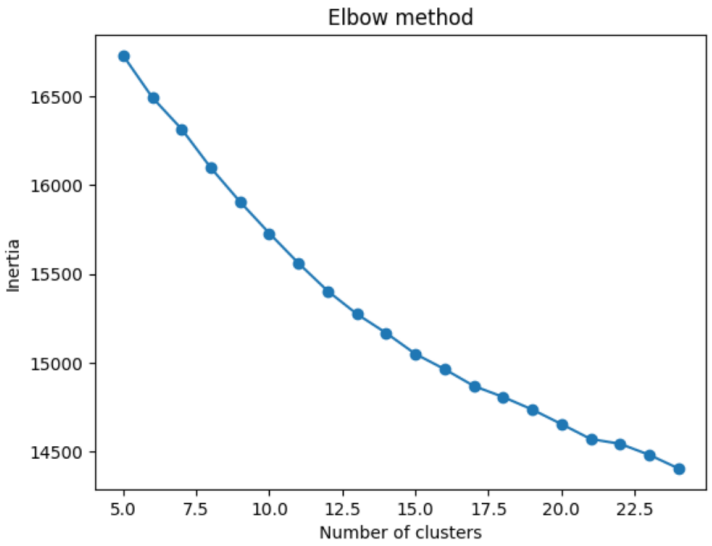

Figure 8: Elbow Method Plot for BERT K-Means Clustering

Figure 9: BERT Embeddings Clustering Word Clouds (20 Clusters)

In addition to normal word frequency clouds, we created representations of full recipe titles each cluster, highlighting the most frequently repeated complete entries. These are displayed in Figure 10.

Figure 10: BERT K-Means Clustering: Word Clouds of Most Frequent Full Titles

Agglomerative clustering was investigated for BERT embeddings; however, due to its $O(n^2)$ memory and $O(n^3)$ time complexity, the algorithm was computationally infeasible even on a reduced subset of 5,000 samples. It failed to complete within an acceptable timeline and was so removed from further consideration.

5.1.3 Category Similarity via BERT Embeddings

In a different technique, we created 13 food categories (e.g., *salad*, *soup*, *dessert*) and computed cosine similarities between each recipe title and the category names, assigning the most similar one. This method produced a well-distributed categorization (Figure 11), with *dessert* being the most prevalent category. Figures 12 and 13 show word clouds based on common keywords and entire title frequencies, respectively. The silhouette score for this method was **0.0260**.

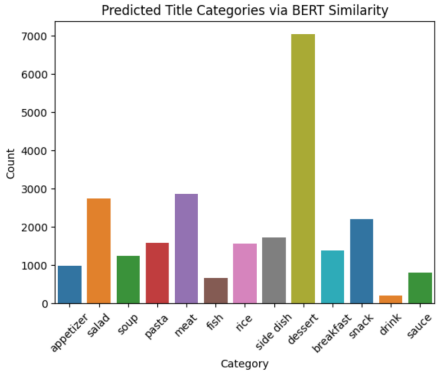

Figure 11: Predicted Title Categories via BERT Similarity

Figure 12: Word Clouds by Predicted Category (Frequent Words)

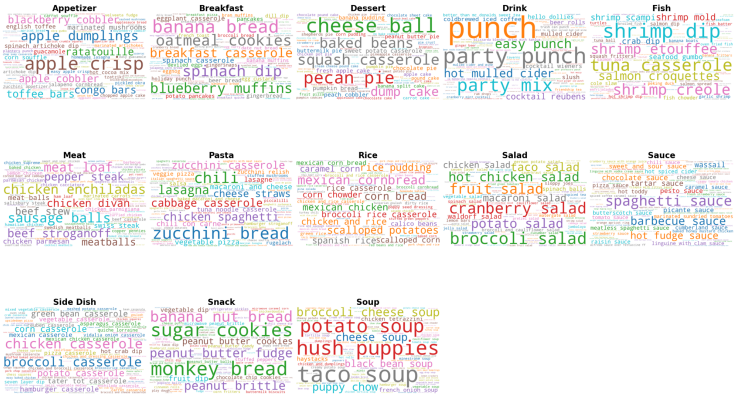

Figure 13: Word Clouds by Predicted Category (Full Title Frequency)

5.2 NER-Based Clustering

5.2.1 BERT Embeddings with K-Means

We clustered recipes based on Named Entity Recognition (NER) data to capture their underlying ingredient structure. A random sample of 25,000 NER entries was retrieved, which were tokenised and embedded using the `all-MiniLM-L6-v2` BERT model. The elbow approach and silhouette analysis as shown in Figure 14 revealed that $k = 15$ was an appropriate value for KMeans clustering.

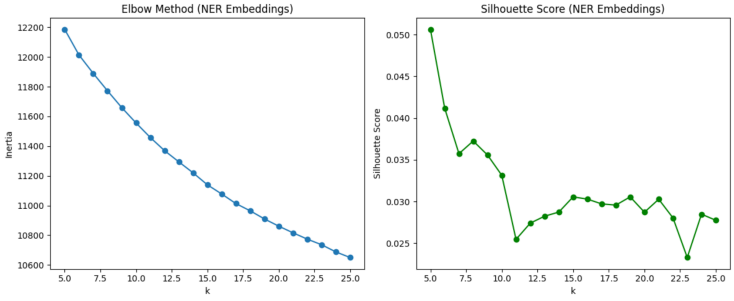

Figure 14: Elbow Method Plot for NER-Based KMeans Clustering

The content of each cluster was visualized using word clouds, created using both ingredient tokens and their corresponding recipe titles (Figures 15, 16). These clusters represented many forms of food, including meat-heavy recipes, baked products, and vegetable-rich dishes.

Figure 15: Word Clouds of BERT KMeans Clusters Based on NER Tokens

Figure 16: Word Clouds of Clustered Recipe Titles (NER-based)

Figure 17 shows a final visualization that highlights the most prevalent recipes per cluster using full-title frequency.

Figure 17: Most Frequent Full Recipe Titles per NER-Based Cluster

The silhouette score for this clustering approach was **0.0306**, which, while modest, suggests significant structure within the NER-based ingredient embeddings.

5.2.2 Category Similarity via BERT Embeddings

We utilized the same method used for title categorization to apply category similarity matching to the NER-based ingredient representations. Each sample was assigned to one of the specified food categories using the label with the highest cosine similarity score.

In order to understand the categorization results, we visualized each category with word clouds. Figure 18 displays the most common words from each NER category's titles, whereas Figure 19 displays the most commonly repeated full recipe titles.

Figure 18: Word Clouds by Category (NER-based, Common Title Words)

Figure 19: Word Clouds by Category (NER-based, Full Titles)

This method provided a silhouette score of **0.0054**, which is lower than other ways, showing poorer separation within clusters while still providing useful high-level labels for describing and filtering the dataset.

6 Fine-tuned Model

6.1 Data Filtering and Category-Based Sampling

Given the large size of the dataset, direct fine-tuning proved computationally expensive. To extract a high-quality, balanced subset, we first embed the normalized NER fields using the `all-MiniLM-L6-v2` model. The embeddings were decreased in dimensionality using Principal Component Analysis (PCA), keeping 80% of the variation.

Next, we used HDBSCAN clustering on the reduced embeddings. This density-based method discovered 28 useful clusters while classifying over 100,000 points as noise. Each cluster was carefully inspected and mapped to a descriptive food category like *Savory Casseroles & Soups*, *Baking & Cake Mix Desserts*, or *Hearty Mains*. Samples labeled general/noise were eliminated from further processing.

To ensure that the generation model received clean and well-structured examples, we used additional filters: only recipes with 4-15 ingredients and at least three instruction stages were kept. This resulted in a chosen subset of 10,531 recipes spanning eight food groups. Table 4 shows the categories and their counts.

Label Category	Count
Baking & Cake Mix Desserts	6,713
Hearty Mains / Meat-Based	2,017
Savory Casseroles & Soups	923
Fruit Jello / Dessert Salads	264
Pie / Crust Desserts	250
Punches & Drinks	201
Cold Layered / Vegetable Salads	128
Appetizer Spreads	35

Table 4: Final category distribution in the filtered dataset

6.2 Training and Test Set Construction

To prepare for model fine-tuning, we divided the curated subset into training and test sets using stratified sampling based on food categories. This resulted in 9,515 training and 500 test samples, ensuring that all main categories were fairly represented in both sets.

Each sample was converted to an instruction-input-output format. The instruction was fixed as *"You are an expert chef. Using the ingredients below, write a delicious food recipe."* The ingredients list was used as the model input, with the associated directions serving as the predicted result.

Finally, the train and test DataFrames were transformed into Hugging Face Datasets to improve compatibility with transformer-based model training. Each sample included three fields: `instruction`, `input` and `output`. After processing, the datasets were stored in `Parquet` format as `train.parquet` and `test.parquet` for easy loading and reuse during training.

6.3 Model Fine-Tuning and Generation

To efficiently train a recipe generation model, we used the Unsloth library to fine-tune various quantised transformer models using Low-Rank Adaptation (LoRA) in 4-bit precision. All models were fine-tuned on a preprocessed recipe dataset using the Alpaca prompt format.

Model Selection and Configuration

We experimented with four base models:

- **Qwen2.5-7B-bnb-4bit**
- **Mistral-7B-bnb-4bit**
- **Gemma-2B-bnb-4bit**
- **TinyLlama-bnb-4bit**

All models were loaded using `FastLanguageModel.from_pretrained()` and patched with parameter-efficient LoRA layers (`r=16`) that target the standard QKV and MLP projection heads. Gradient checkpointing was set to save memory use.

Prompt Formatting

We used the Alpaca instruction-following template, which combines the `instruction`, `input`, and `output` variables into one prepared string. To avoid overgeneration, each prompt concluded with the model's EOS token.

```
Below is an instruction that describes a task,
paired with an input that provides further
context.  Write a response that appropriately
completes the request.
```

```
### Instruction: You are an expert chef. Using
the ingredients below, write a delicious food
recipe.
### Input: spaghetti, eggs, parmesan cheese,
pancetta, black pepper, salt, olive oil
### Response: ...
```

Training Setup

Fine-tuning has been performed with the `SFTTrainer` from HuggingFace's TRL, using the following parameters:

- Batch size: 2 (with gradient accumulation of 4)
- Epochs: 1
- Learning rate: 2×10^{-4}
- Optimizer: `adamw_8bit`
- Precision: FP16 or BF16 (depending on hardware)

The models were trained for one epoch and saved locally with the `model.save_pretrained()` method for later on inference.

Inference Results

Following training, we assessed each model's generation quality by providing short ingredient lists. Here are some samples of generated responses for each fine-tuned model:

Qwen2.5-7B-bnb-4bit

- **Input:** `bread, ham, cheese`
  Output: *Spread bread with butter, layer ham and cheese, roll up and slice, bake at 350°F for 20 minutes.*
- **Input:** `pasta, minced meat, cheese`
  Output: *Cook pasta according to package directions, drain and rinse. In a large bowl, combine pasta, meat, egg, salt, pepper and cheese, mix well. Spoon into greased baking dish, sprinkle with remaining cheese, bake at 350°F for 30 minutes. Serve with spaghetti sauce.*
- **Input:** `eggs, flour, sugar, milk, butter, vanilla extract, baking powder, salt`
  Output: *Beat eggs, add sugar, milk, butter and vanilla. Sift flour, baking powder and salt. Add to mixture, pour into greased and floured pan, bake at 350°F for 1 hour.*
- **Input:** `spaghetti, eggs, parmesan cheese, pancetta, black pepper, salt, olive oil`
  Output: *Cook spaghetti in boiling salted water until al dente, rinse with cold water. In a bowl, beat eggs, add parmesan, pancetta, pepper, and salt. Toss with spaghetti and serve immediately.*

Mistral-7B-bnb-4bit

- **Input:** `spaghetti, eggs, parmesan cheese, pancetta, black pepper, salt, olive oil`
  Output: *Cook spaghetti according to package directions, rinse with cold water. Place in a bowl with eggs, parmesan, pancetta, pepper, and salt. Toss to combine. Heat oil in a skillet and cook mixture until golden brown, about 5 minutes. Serve immediately.*

TinyLlama-bnb-4bit

- **Input:** `ham, bread, cheese`
  Output: *Ham, bread, cheese. Ham, bread, cheese. Ham, bread, cheese...*
  (The model entered a repetition loop, indicating poor convergence or instruction-following alignment.)

Gemma-2B-bnb-4bit

- **Input:** `spaghetti, eggs, parmesan cheese, pancetta, black pepper, salt, olive oil`
  Output: *Cook spaghetti according to package directions, drain and set aside. Beat eggs, add parmesan, pancetta, pepper, and salt. Toss with spaghetti and bake in a dish at 350°F for 20 minutes. Serve hot.*

Qwen2.5-3B-bnb-4bit

- **Input:** `spaghetti, eggs, parmesan cheese, pancetta, black pepper, salt, olive oil`
  Output: *Cook spaghetti according to package directions, rinse with cold water. In a large bowl, combine spaghetti, eggs, parmesan, pancetta, pepper, and salt. Mix well. Heat olive oil in a skillet, add spaghetti mixture and cook until heated through, about 5 minutes. Serve immediately.*

These results show that different models function differently. Some models (like TinyLlama) had trouble with task adherence or repetition, but the majority generated recipes that were structured and logical. Fluent and task-relevant instructions were consistently produced by the larger Qwen and Mistral variants.

6.4 Evaluation

Automatic Evaluation Metrics

Using the same held-out test set of 500 recipes, we assessed each model using five automatic metrics:

- **BLEU**: Measures n-gram overlap between generated and reference recipes.
- **ROUGE-1 / ROUGE-2 / ROUGE-L**: Captures unigrams, bigrams, and longest matching sequences respectively.
- **BERTScore (F1)**: Embedding-based semantic similarity between generated and reference texts.

According to the results in Table 5, optimised models routinely perform better than their pre-trained counterparts on every criterion.

Inference Time vs. Semantic Quality

Figure 20 demonstrates trade-offs between BERTScore and inference time. Models like **Gemma-2B-finetuned** strike an ideal compromise between high semantic similarity and low latency. TinyLlama models, on the other hand, are slow and inefficient.

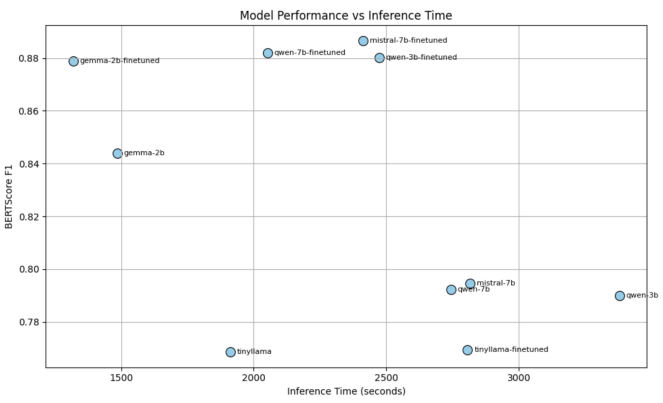

Figure 20: Model Performance vs Inference Time (BERTScore F1)

Table 5: Evaluation Metrics and Inference Time per Model

Model	BLEU	ROUGE-L	ROUGE-2	ROUGE-1	BERTScore (F1)	Inference Time (s)
qwen-7b	0.0173	0.1369	0.0357	0.2254	0.7923	2744.07
mistral-7b	0.0215	0.1374	0.0387	0.2193	0.7946	2815.89
tinyllama	0.0093	0.1424	0.0276	0.1758	0.7687	1910.99
gemma-2b	0.0594	0.2106	0.0749	0.3126	0.8440	1485.63
qwen-3b	0.0141	0.1384	0.0324	0.2215	0.7900	3380.46
qwen-7b-finetuned	**0.1363**	0.3317	0.1737	0.4212	0.8819	2052.41
mistral-7b-finetuned	**0.1512**	**0.3598**	**0.1994**	**0.4481**	**0.8865**	2412.78
tinyllama-finetuned	0.0098	0.1465	0.0291	0.1797	0.7694	2807.10
gemma-2b-finetuned	0.1248	0.3193	0.1637	0.4051	0.8788	**1318.68**
qwen-3b-finetuned	0.1329	0.3232	0.1718	0.4125	0.8801	2473.46

Finetuning Gains

We found the percentage improvement from pre-trained to fine-tuned models using all five metrics. Figure 21 clearly reveals large boosts—especially in BLEU and ROUGE-2—with **Mistral-7B** and **Qwen-3B** improving over 800% and 430% in BLEU, respectively.

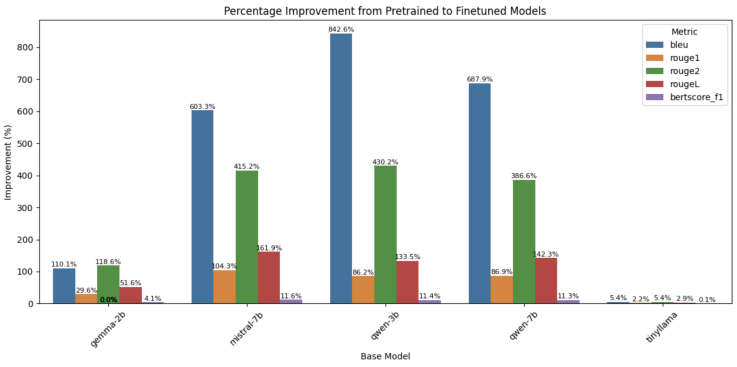

Figure 21: Percentage Improvement from Pretrained to Fine-tuned Models

Overall Ranking

To balance performance and efficiency, we normalised BLEU, ROUGE-L, BERTScore, and speed. Figure 22 demonstrates that **Gemma-2B-finetuned**, **Mistral-7B-finetuned**, and **Qwen-7B-finetuned** scored best overall, making them top choices for deployment.

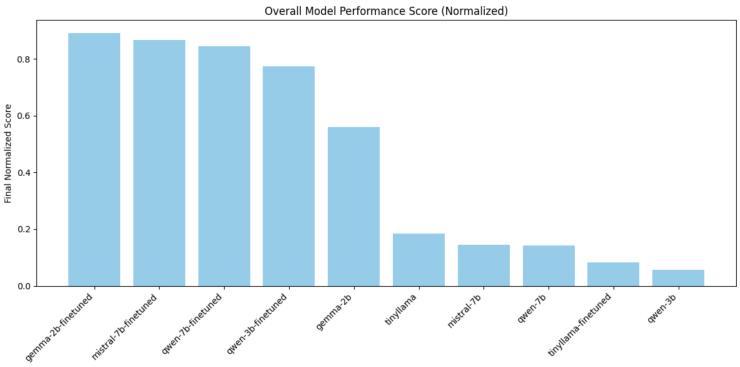

Figure 22: Overall Model Performance Score (Normalized)

Human Evaluation

A qualitative human review on a random sample of 15 test recipes was conducted using the best-performing model, **Gemma-2B-finetuned**. ChatGPT was instructed to compare created recipes to their ground-truth references on:

- Procedural correctness
- Ingredient usage
- Fidelity to reference content

Each response was graded on a scale of 1 to 10. The average score was **7.2/10**, showing great fluency and accuracy, with slight restrictions in managing complex cooking specifics such as boiling times or step order. A few samples suffered from under-generation or simplification.

Discussion

Fine-tuning improved output quality significantly across all models, with some gaining over 600% in BLEU. While larger models (e.g., Mistral, Qwen) gave slightly more accurate outputs, the **Gemma-2B-finetuned** model stood out for its:

- Low inference time (1318s vs 2400s+ for others)
- Competitive performance (BERTScore F1: 0.8788)
- Small size and memory efficiency

Thus, for actual recipe-generation applications where inference time and hardware restrictions matter, **Gemma-2B-finetuned** represents the optimal trade-off.

7 Limitations

This study has several important limitations that may affect the generalizability and completeness of the results. First, the automatic metrics used—particularly BLEU and ROUGE—are sensitive to surface-level overlap and may unfairly penalize valid generations that differ stylistically or structurally from the reference texts. Second, the final performance score aggregates normalized values from all metrics using equal weighting; while this offers simplicity, it does not account for task-specific priorities, such as placing greater emphasis on semantic quality or efficiency. Third, inference was constrained by hardware limitations (a single NVIDIA T4 GPU) and limited training time, which required running models with small batch sizes and using a reduced dataset. These constraints may have negatively impacted both fine-tuning quality and model generalization. Finally, although finetuning was performed using the Unsloth library for consistency and speed, this study did not systematically explore how different fine-tuning settings—such as training duration, data volume, or learning rate—might influence final model performance.

8 Conclusion

This study showed that employing improved large language models for recipe generation is both possible and useful. We began with a huge, heterogeneous dataset of more than 2 million recipes, which we then carefully cleaned, enriched, and categorized using a variety of NLP approaches, such as normalization, clustering, and LLM-based nutritional estimate.

We used LoRA to fine-tune several quantised models, assessed them quantitatively and qualitatively, and compared them to five automatic metrics. The findings demonstrated that fine-tuning greatly enhances output quality, with models such as **Mistral-7B** and **Qwen-3B** achieving BLEU score improvements of over 800%. But the **Gemma-2B-finetuned** model was the best overall performer, combining short inference time (1318s), small model size, and high semantic similarity (BERTScore F1: 0.8788), which makes it perfect for real-world applications where efficiency is important.

Despite restrictions in hardware, training time, and evaluation scope, this study demonstrates the practical application of LLMs in food technology and smart cooking helpers. Future plans include adding user choices, enabling multilingual generation, and expanding the system to include voice-controlled interaction or multimodal food assistance. *Smart Chef* advances personalised, AI-driven culinary innovation by integrating structured food knowledge with generative models.

References

[1] M. Bień, M. Gilski, M. Maciejewska, W. Taisner, D. Wisniewski, and A. Lawrynowicz, "RecipeNLG: A cooking recipes dataset for semi-structured text generation," in *Proceedings of the 13th International Conference on Natural Language Generation*, (Dublin, Ireland), pp. 22–28, Association for Computational Linguistics, Dec. 2020.