Applications of NLP in Recipe Texts

Student Names	**Roll Numbers**
Neelu	2022318
Gurupriya Vaikundam	2022191
Rituj Upadhyay	2020570

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering
on 23/04/2025

BTP Track: Research project

BTP Advisor:
Dr. Ganesh Bagler

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

We hereby declare that the work presented in the report entitled **"Applications of NLP in Recipe Texts"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under the guidance of **Dr. Ganesh Bagler**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

Place & Date: New Delhi, 23/04/2025

Neelu Gurupriya Vaikundam Rituj Upadhyay

Certificate

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Ganesh Bagler

Place & Date: New Delhi, 23/04/2025

Abstract

We present a comprehensive study on Applications of NLP in Recipe Texts, focusing on automatic classification of recipes into meal categories (Breakfast, Lunch, Dinner) via text embeddings. Starting with the collection of a 50,000-recipe corpus scraped from sites like The Spruce Eats, Archana's Kitchen, and Food.com, we apply rigorous cleaning, deduplication, and consolidation into 20 standardized categories. Exploratory Data Analysis uncovers ingredient and preparation-time trends across categories. We compare multiple embedding-based pipelines: (i) a Doc2Vec + classical-ML framework (Logistic Regression, SVM, Random Forest, etc.), (ii) a shallow CNN-BiGRU model and its hybrid extension with multi-head attention and auxiliary features, and (iii) transformer-based baselines using frozen BERT embeddings and a fine-tuned RecipeBERT. Our best configuration—RecipeBERT fine-tuned for classification—achieves nearly 60 percent accuracy, significantly outperforming traditional methods. We conclude by discussing practical deployment considerations and outline future work in domain-specific model adaptation and recommendation systems.

Keywords: natural language processing; recipe classification; text embeddings; Doc2Vec; CNN; BiGRU; multi-head attention; BERT; RecipeBERT; machine learning; data cleaning; exploratory data analysis

Acknowledgments

We would like to express our sincerest gratitudes to our advisor, Prof. Ganesh Bagler, for his technical advices, continous support, and valuable feedbacks throughout this work. His helpful suggestions and guidances has been crucial in shaping both the direction and executions of our reseach. We also very thankful to the faculty members at IIIT Delhi for the nicely-structured courses in Data Science and NLP, which has helped us gain initial knowledge that was necessary for starting this work. Big thanks also goes to our friends and collegues, whose thought-provoking discus- sions and sharp critics helped us look at the problem from multiple angles and added many insights into the project. At last, we want to specially thank our families for their patiency, encouragment, and always being emotionally there for us during the ups and downs of this whole journey. This project wouldn't have been done without the combined efforts of all those who stood by us, and we are truly gratful for their impacts on our works.

Work Distribution

Task	**Team Member**
Dataset scraping & construction	All members collaboratively
Data cleaning & standardization	All members collaboratively
Category consolidation & EDA	All members collaboratively
Doc2Vec embedding & classical ML experiments	Gurupriya Vaikundam
CNN-BiGRU baseline & hybrid model development	Rituj Upadhyay
BERT baseline & RecipeBERT fine-tuning	Neelu
Report writing & figure/table generation	Gurupriya Vaikundam, Rituj Upadhyay
Project coordination & supervision	Dr. Ganesh Bagler (Guide)

Contents

Chapter 1

Introduction

In recent years, the proliferation of online recipe repositories has created vast corpora of cooking instructions, ingredient lists, and metadata ripe for Natural Language Processing (NLP) analysis. Automated understanding of recipe texts holds promise for personalized meal recommendations, dietary planning, and culinary trend analysis. However, recipe language is heterogeneous—mixing procedural instructions, domain-specific vocabulary, and varying formats—which poses challenges for traditional NLP pipelines.

This work explores end-to-end methods for classifying recipes into broad meal categories (Breakfast, Lunch, Dinner), using only textual data. We begin by constructing a large, diverse dataset via web scraping and apply systematic cleaning and category consolidation to yield 20 standardized labels. We then investigate a spectrum of embedding techniques and classifiers: from Doc2Vec combined with classical machine-learning algorithms, through convolutional and recurrent neural architectures with attention, to modern transformer-based models including a fine-tuned RecipeBERT.

Our experiments demonstrate that domain-specific pre-training and fine-tuning (RecipeBERT) significantly outperform lighter-weight models, achieving nearly 60 percentage classification accuracy. We also analyze the interpretability and deployment trade-offs of each approach. Finally, we outline directions for future work, such as extending to ingredient-level recommendations, multilingual recipe corpora, and integration within real-time cooking assistants.

Chapter 2

Dataset Acquisition and Preliminary Analysis

This chapter outlines the complete method which we has used to collect, clean up, and analyse the recipe dataset in our project. The datas was scraped from different online recipe websites, like The Spruce Eats, Archana's Kitchen and also Food.com. Our main goal were to build a big and diverse dataset having both structured as well as unstructured informations about thousands of recipes, which can be used later for tasks like classification or recommendations.

2.1 Web Scraping and Dataset Construction

The initial step involved scraping recipes using Selenium and BeautifulSoup. Selenium was chosen for its ability to interact with dynamically rendered JavaScript content, which is common on modern websites. For example, in the case of The Spruce Eats, recipe pages were loaded using pagination, and individual cards required simulated clicks to reveal detailed content. XPath and CSS selectors were used to identify elements like the recipe name, ingredients, instructions, region or cuisine, preparation time, servings, and tags.

To handle dynamic pagination, our script incremented page numbers in the URL (https://www.thespruceeats.com/search?page=2...) and extracted all recipe URLs on each page. After navigating to the individual recipe page, BeautifulSoup was used to parse the HTML source and extract relevant fields. Each recipe was saved with fields such as:

Recipe Name
List of Ingredients
List of Instructions
Preparation Time

Servings

Region/Cuisine

Tags (for category identification)

More scraping was done for Archana's Kitchen, where the serving info is usually hided under the "Makes" part, and also for Food.com, where the datas were only partly structured and not fully consistent.

2.2 Data Cleaning and Standardization

Post-scraping, the dataset had over 26,000 recipes. However, several inconsistencies and redundancies required rigorous data cleaning:

Unifying Text Format: Many columns such as ingredients and categories had values stored in multiple formats—some as comma-separated strings, others as Python-style lists. These were converted to uniform Python lists.

Removing Non-English Recipes: Recipes not in English were identified by checking for non-ASCII characters in recipe names and dropped.

Handling Null Values: Columns such as preparation time, servings, and category had missing entries. We removed rows with essential fields like instructions or recipe names missing and treated sparse columns accordingly. The "Calories per Serving" column, with over 40,000 nulls, was entirely dropped.

Duplicate Handling: We tried to find duplicate recipes by looking at combination of recipe_name, ingredients and also the instructions. After removing the duplicates, we made sure the list-based columns had same kind of formatting so that they can be hashed properly and be used more easily for next analysis.

Standardizing Preparation Time: Preparation time values existed in various formats—minutes, hours, mixed. We implemented a parser that extracted hours and minutes and converted all values into a total duration in minutes.

Ingredient Cleaning: Ingredient lists were cleaned using a custom parser. This involved:

Removing common measures (e.g., tsp, cup, gram)

Removing stopwords and filler adjectives (e.g., "fresh", "extra")

Lemmatizing ingredient names to their root forms

Removing quantities and units

This process helped reduce noise and redundancy in the ingredients used for classification and visualization.

2.3 Category Consolidation

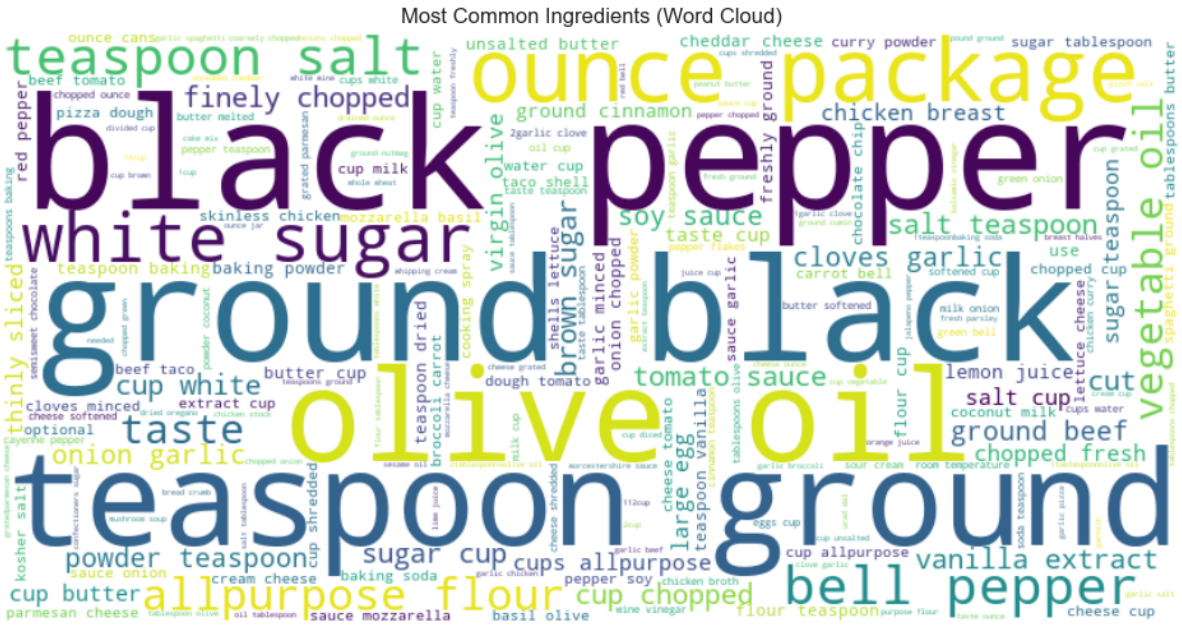
The original dataset had more than 600 unique recipe categories, but many of them were kind of repeating or overlaped each other. To deal with this and make things more uniform in the dataset, we did a consolidation step. First we taken out the top 30 most frequent categories and analysed them more closely. A manual mapping strategy was then employed to unify semantically similar subcategories under broader, more interpretable labels such as "Dessert", "Main Dish", "Appetizer", and others. Categories that were either too rare or too ambiguous to interpret confidently were grouped into a generic "Other" category. This approach helped reduce label noise and simplify the downstream classification task. To check how strong our manual mapping was, we also used a unsupervised clustering method. We did TF-IDF vectorising on the category names and then applied KMeans clustering, which shown patterns that kind of matched with our manual groups. This helped in making the mappings more believable and understandable. This two step consolidation gave us a much cleaner set of 20 clear categories, which made the class distribution more balanced and also made the insights more easier during exploring and modelling parts.

2.4 Exploratory Data Analysis (EDA)

To better understand the underlying patterns and structure of the cleaned dataset, several exploratory data analysis (EDA) steps were performed. These analysis gave us useful insights which helped us later when we were deciding on how to do feature engineering and model building.

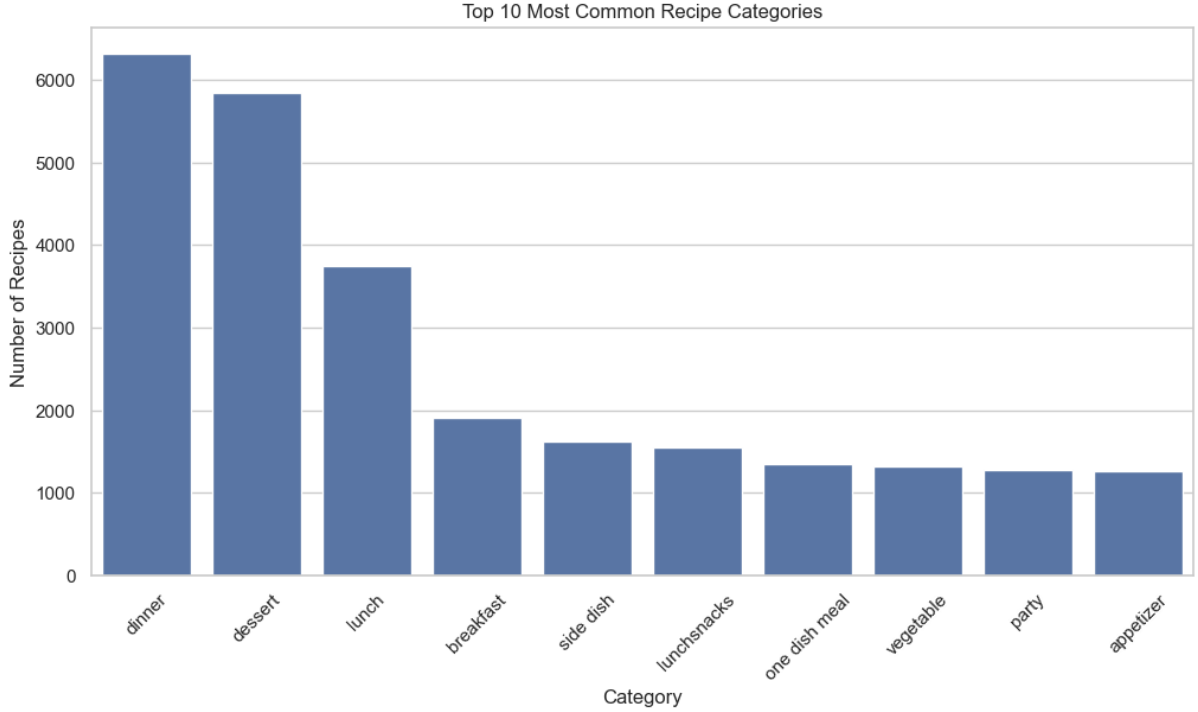
2.4.1 Ingredients Analysis

One of the key areas of investigation involved analyzing the ingredients used across recipes. Cleaned ingredient tokens were visualized using word clouds to highlight the most commonly used items, which included ubiquitous ingredients such as salt, oil, and garlic. In addition, bar plots were created to quantify the frequency of specific ingredients. A heatmap was also generated to illustrate the number of unique ingredients utilized within each of the 20 main categories. This analysis showed some interesting trends, like we noticed that dessert recipes usually needs more variety of ingredients on average when we compare them to easier categories like snacks or drinks.



2.4.2 Category Distribution

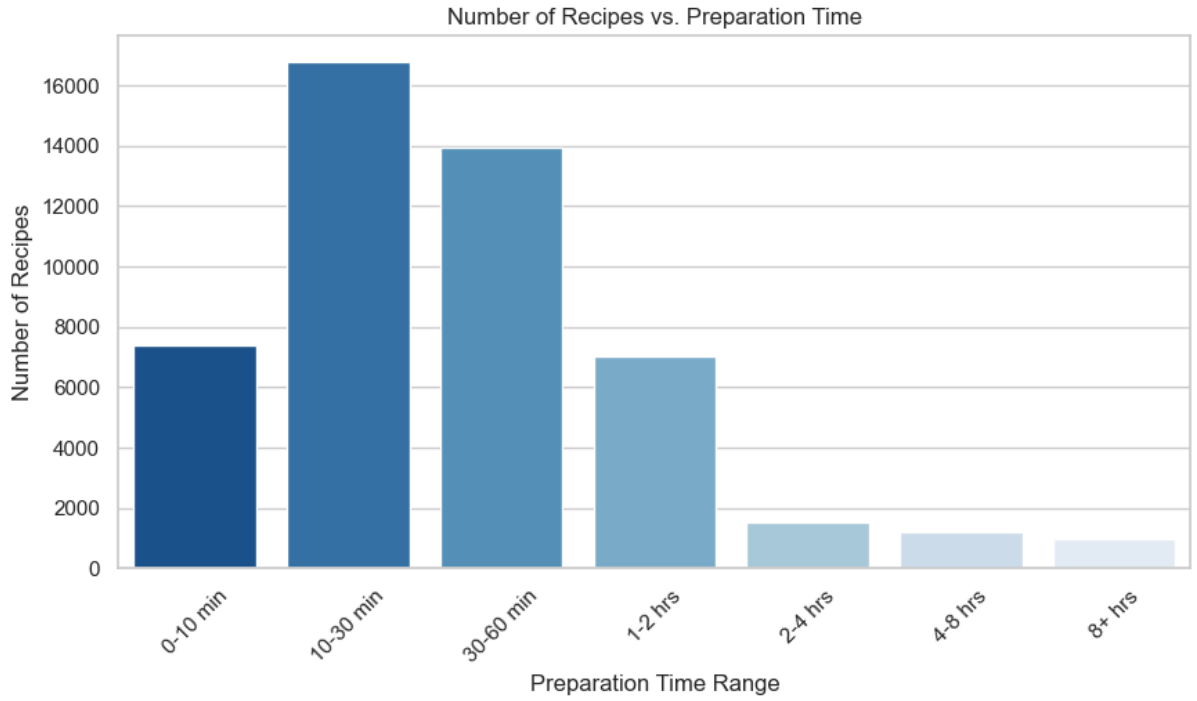
We made a bar chart to show how the 20 merged categories are spread across the dataset. This helped us see that some categories were appearing way more often than others. The top five categories that showed up the most were “Dessert”, “Main Dish”, “Appetizer”, “Breakfast” and “Side Dish”. Knowing about this imbalance was important so we could plan class-balanced training and also understand the performance numbers better while doing classification.

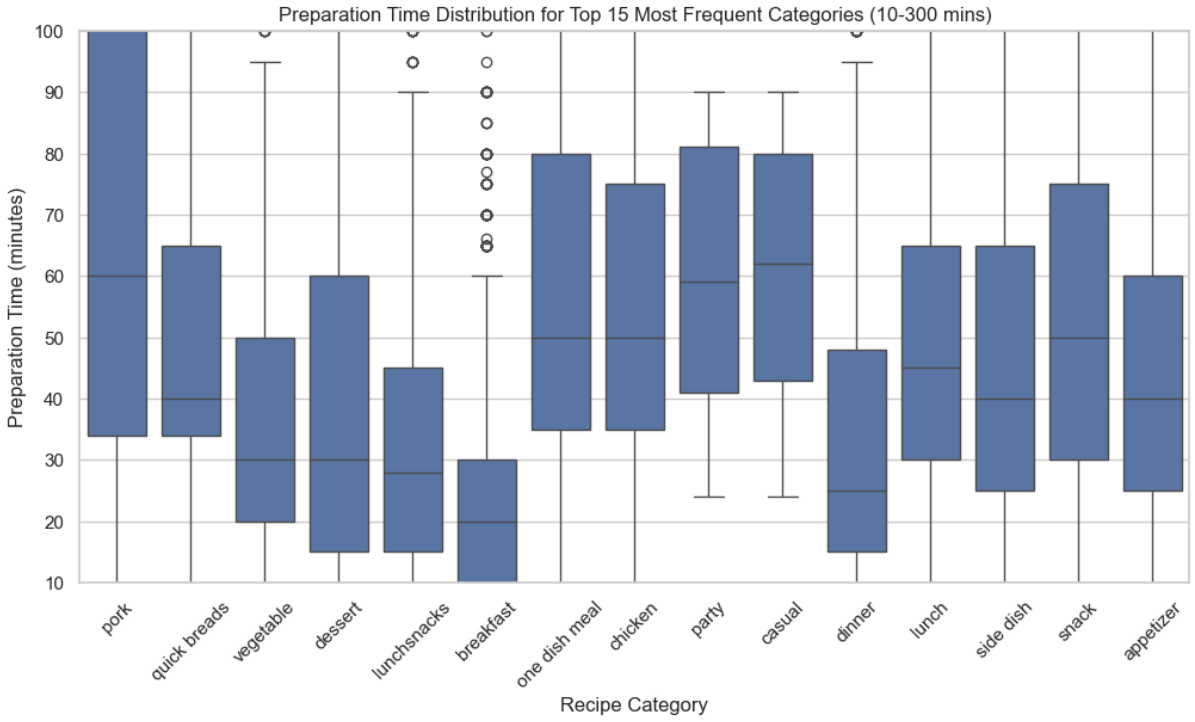


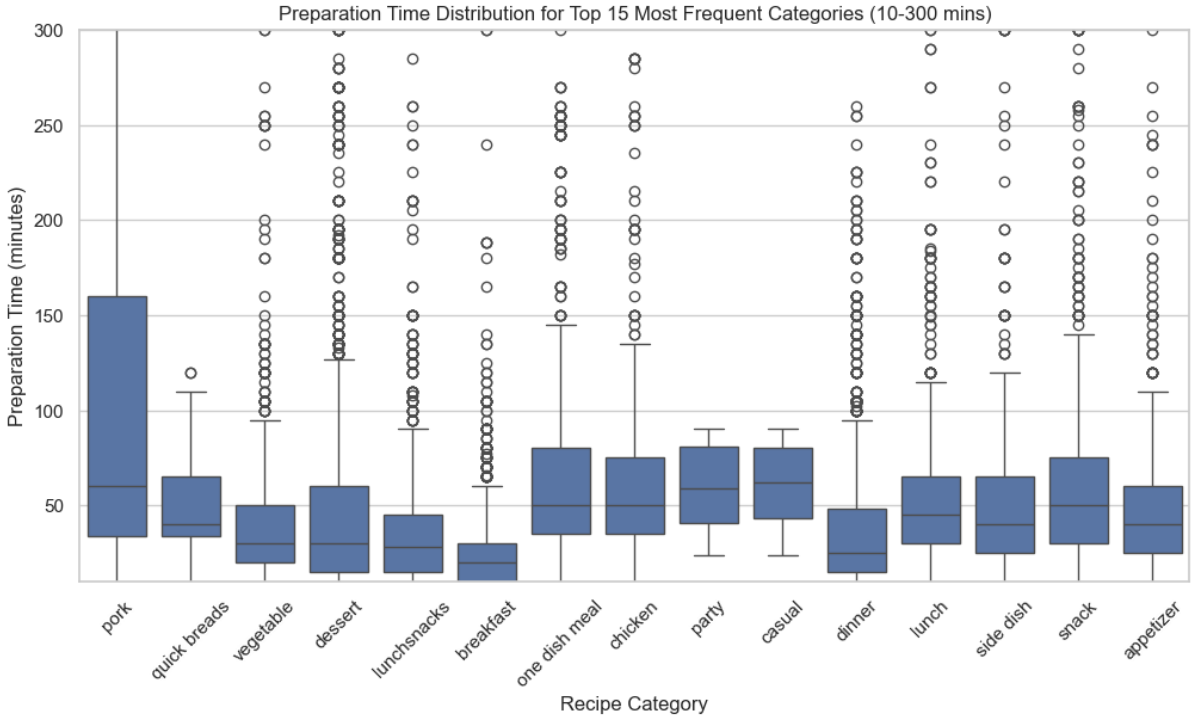
2.4.3 Preparation Time Insights

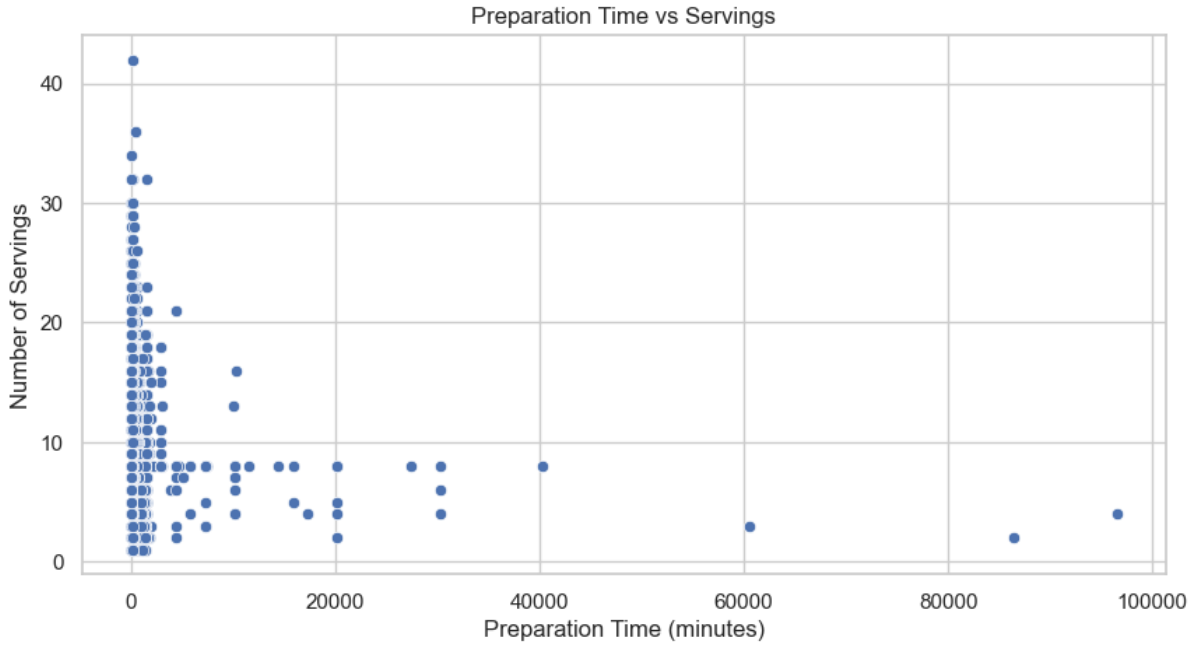
The Preparation Time column was changed into number format, so that we can do more statistical and visual analysis on it. A histogram of the prep times showed that most of the recipes takes around 15 to 45 minutes to make. To look deeper into this, we also used box plots to compare how prep time differ between the various categories.

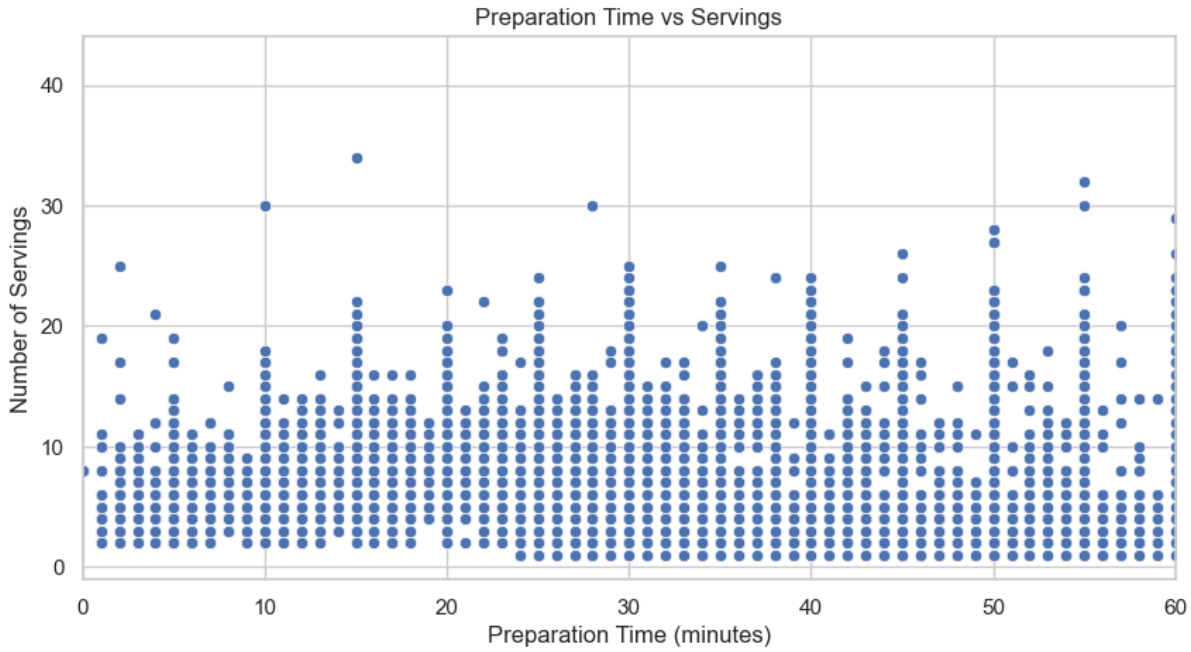
This analysis highlighted clear differences in cooking complexity among food types. Outliers were also identified, particularly those recipes with excessively long preparation times exceeding six hours. These could optionally be excluded to reduce noise in training and evaluation processes.




Preparation Time Distribution for Top 15 Most Frequent Categories (10-300 mins)
Preparation Time (minutes)
100
90
80
70
60
50
40
30
20
10
pork
quick breads
vegetable
dessert
lunchsnacks
breakfast
one dish meal
chicken
party
casual
dinner
lunch
side dish
snack
appetizer
Recipe Category

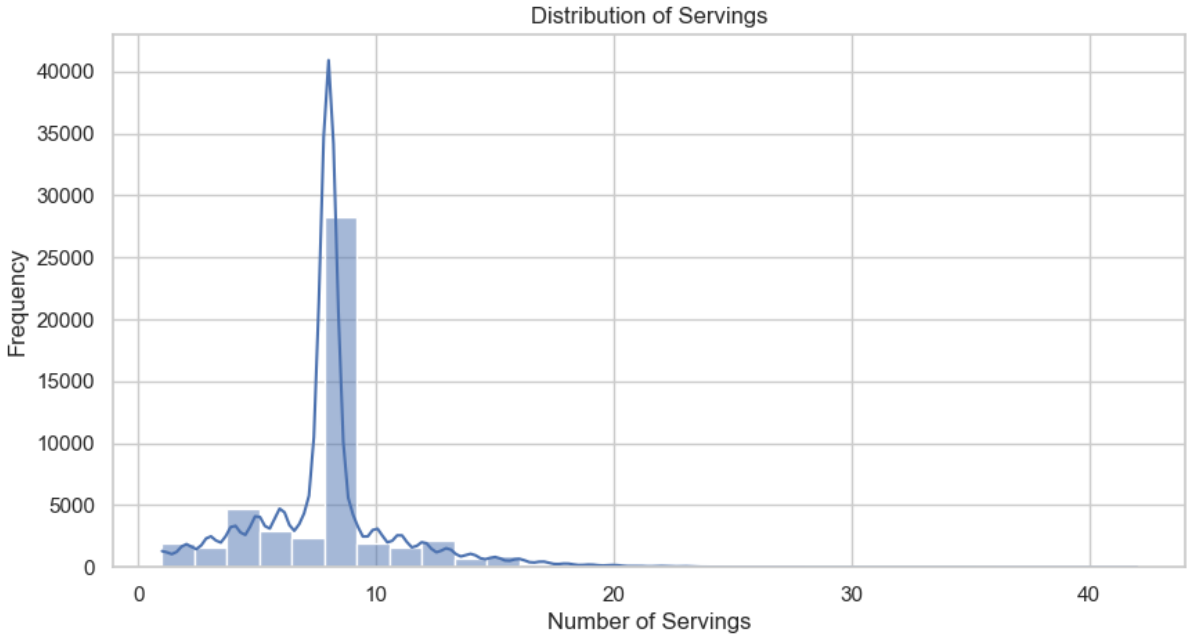

Preparation Time Distribution for Top 15 Most Frequent Categories (10-300 mins)
Preparation Time (minutes)
300
250
200
150
100
50
pork
quick breads
vegetable
dessert
lunchsnacks
breakfast
one dish meal
chicken
party
casual
dinner
lunch
side dish
snack
appetizer
Recipe Category

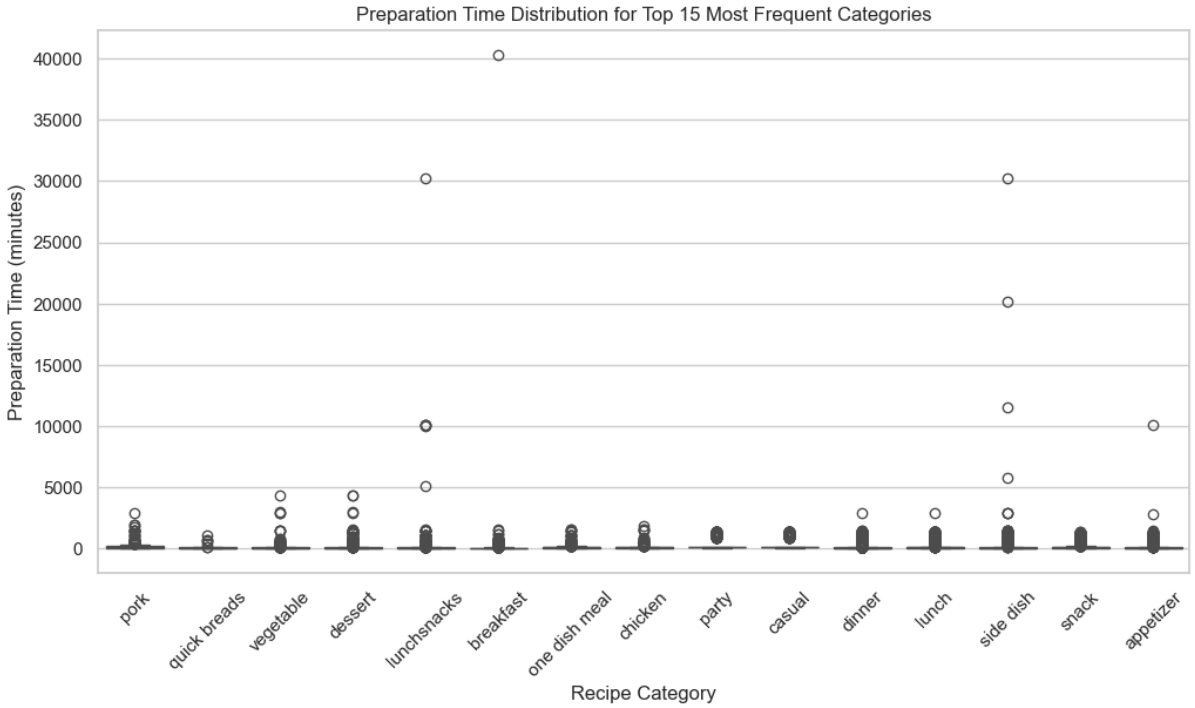




2.4.4 Region/Cuisine Trends

The Region/Cuisine field, though auxiliary, was analyzed to observe geographical trends in the data. Visualizations showed that cuisines labeled as "American", "Indian", and "Italian" were among the most represented in the dataset. However, because a significant portion of records either lacked this field or had inconsistent labels, the region/cuisine information was ultimately used as a supplementary metadata field rather than a core input for modeling.





2.5 Final Dataset Summary

After completing the full preprocessing pipeline—including data cleaning, normalization, and consolidation—the final dataset was significantly more structured and ready for downstream modeling tasks. The cleaned dataset contained approximately 50,000 unique, deduplicated recipes. The number of distinct categories was reduced and standardized to 20 major groups, improving class balance and interpretability. Preparation times were normalized to integer values representing minutes, and ingredient lists were cleaned, lemmatized, and parsed into uniform formats. All categorical fields were either

mapped or encoded appropriately for use in machine learning models. Instruction texts were also cleaned properly, normalised, and lemmatised so that it can be used as good input for different NLP models. In the end, the dataset we got was strong, clean and pretty much ready for training high-level text classification models.

Chapter 3

Baseline Model: BERT Embeddings with Logistic Regression

As a starting point for the BTP project on recipe classification, a baseline model using pre-trained BERT embeddings was implemented. The textual content — recipe names, ingredients, and instructions — were encoded using BERT, and passed into a logistic regression classifier to predict the recipe category. This model aimed to establish a working pipeline for text preprocessing, feature extraction, and model evaluation. The goal was not high performance but to test integration and spot initial limitations.

3.1 Text Vectorization Using BERT

Text from recipes was tokenized and passed through a frozen pre-trained BERT model. The [CLS] token from the last hidden state served as the vector representation of each recipe. This fixed-size embedding was then used directly as input features for the classifier. Since no fine-tuning was applied, the model lacked understanding of culinary-specific vocabulary and semantics.

3.2 Feature Concatenation and Final Representation

The model used a single [CLS] token embedding for each recipe text, without any concatenation of different recipe components. This representation served as the only input to the logistic regression model, limiting expressiveness.

3.3 Classifier Architecture and Training

A simple logistic regression classifier was trained on top of the BERT embeddings. The architecture consisted of a single linear layer without any hidden layers or non-linear

transformations. No fine-tuning of BERT was performed. The training loss steadily decreased from 6.52 at epoch 0 to 2.78 by epoch 90, indicating correct optimization despite the model's simplicity.

3.4 Evaluation, Results And Conclusion

Test Accuracy: 30.10Loss Trend: Smooth and steady decline in loss throughout training Observations: Model did not overfit; it learned gradually and showed stable convergence. Limitations arose from using a non-fine-tuned model, a weak classifier, and class imbalance in the dataset.

The baseline model effectively established the foundational pipeline for recipe classification. It helped identify major challenges such as the need for domain adaptation, class imbalance, and limited expressiveness of simple classifiers. Despite modest performance, it served as a critical benchmark and gave clear direction for deeper modeling strategies such as using domain-specific transformers and fine-tuning methods.

Chapter 4

Text-Based Classification Model for Recipe Categorization Using RNN

This section presents an account of the models experimented with for predicting recipe categories using only the textual cooking instructions. Our objective was to explore how instructional text, once cleaned and properly encoded, could be leveraged to infer broad-level categories such as "Dessert", "Main Dish", or "Appetizer". Two significant architectures were created and tested: a vanilla CNN-BiGRU model, which was the baseline, and an improved hybrid model that dramatically enhanced performance by leveraging architectural and preprocessing advancements.

4.1 Baseline Model: CNN + BiGRU

We started with a basic baseline model consisting of a convolutional neural network (CNN) followed by a bidirectional gated recurrent unit (BiGRU). This was driven by the local n-gram-like features that CNNs can capture, and the BiGRU to model sequential dependencies in the instructional text.

The model input was tokenized cooking instructions, truncated to 200 tokens and embedded with a trainable 96-dimensional layer.

Model Architecture includes:

- CNN Branch: A single Conv1D layer with ReLU activation followed by global max pooling.
- BiGRU Branch: Bidirectional(GRU(64)) with return_sequences=True, followed by GlobalMaxPooling1D.
- Fusion: Concatenation of CNN and BiGRU outputs.
- Dense Layers: One fully connected layer (128 units) followed by dropout.

- Output Layer: softmax layer for multi-class classification.

Although this model achieved modest performance (46–48% validation accuracy), it had no attention mechanisms and was not utilizing auxiliary information like instruction length, which was determined to be prognostic when exploring the data.

In spite of its weakness, this baseline did set a helpful benchmark for performance and indicated the promise of BiGRU architectures in learning sequential relationships in recipes.

4.2 Final Model: Hybrid BiGRU-CNN with Attention and Auxiliary Features

To build on the baseline, a more robust hybrid architecture was created, incorporating not just CNN and BiGRU layers but also multi-head attention, instruction length metadata, and a more advanced preprocessing pipeline.

Preprocessing Pipeline:

- Instruction Cleaning: Instructions were cleaned to eliminate special characters, normalize units (e.g., °F, tbsp), and convert measurement formats to standardized tokens.
- Lemmatization Tokenization: Instructions were tokenized and lemmatized, paying special care to frequent cooking verbs (e.g., "chopped" → "chop").
- Stopword Filtering: Core stopwords were filtered out, preserving contextually significant terms such as "until" and "when".
- Instruction Length Feature: A manually crafted feature for the number of tokens in every instruction was extracted and normalized to be integrated into the model later.
- Label Encoding: Recipe classes were manually encoded to 20 large classes and then encoded with LabelEncoder.
- Sequence Padding: Sequences of tokens were padded to 95th percentile length (200 tokens) for standardized input in batches.
- These preprocessing operations aided in noise reduction and ensured meaningful input representation for the training of deep learning models.

The hybrid architecture had three significant branches:

- A BiGRU and multi-head attention, enabling the model to attend to salient portions of the instruction sequences and to contextualize weighting of tokens.
- A multi-kernel CNN branch, capturing diverse local n-gram patterns using convolution filters with sizes 2, 3, and 5.
- A dense branch managing the normalized instruction length, giving the model a signal regarding the complexity or simplicity of the recipe.

Model Architecture includes:

- Input 1 – Instruction Sequences → Embedding Layer (192-dimensional vectors, mask_zero=True) → SpatialDropout1D(0.4)
- Branch 1: Bidirectional GRU(96 units) → MultiHeadAttention(num_heads=4, key_dim=64) → Residual Connection + Layer Normalization → GlobalMaxPooling1D
- Branch 2: Conv1D layers with kernel sizes 2, 3, 5 (96 filters each, ReLU) → Concatenation → GlobalMaxPooling1D
- Branch 3: Length Input → Dense(32 units, ReLU)
- Final Merging → Batch Normalization → Dense(256, ReLU) + Dropout(0.5) → Dense(128, ReLU) + Dropout(0.4) → Output (softmax)

Training used class weighting, early stopping, and ReduceLROnPlateau. The hybrid model achieved 52.25% validation accuracy compared to 48% for the baseline, demonstrating the effectiveness of multi-branch architectures and attention mechanisms.

The comparison of performance between the baseline BiGRU model and the hybrid BiGRU-CNN model demonstrates an obvious jump in classification accuracy with the hybrid model. The 4% increase in validation accuracy indicates that embracing multiple architectural branches as well as other features (such as instruction length and attention mechanisms) results in an expressive model that is able to capture the semantic variations in cooking instructions more effectively.

Although the gain may seem small, in a multi-class text classification problem with noisy real-world data such as recipes, such an improvement is significant and useful in practice, particularly since the model is still computationally tractable.

4.3 Result

To assess the performance of the enhanced hybrid model, we calculated the F1-score for every recipe category with the test dataset. F1-score is a balanced measure that considers

both precision and recall and hence apt for multi-class classification where class imbalance is an issue.

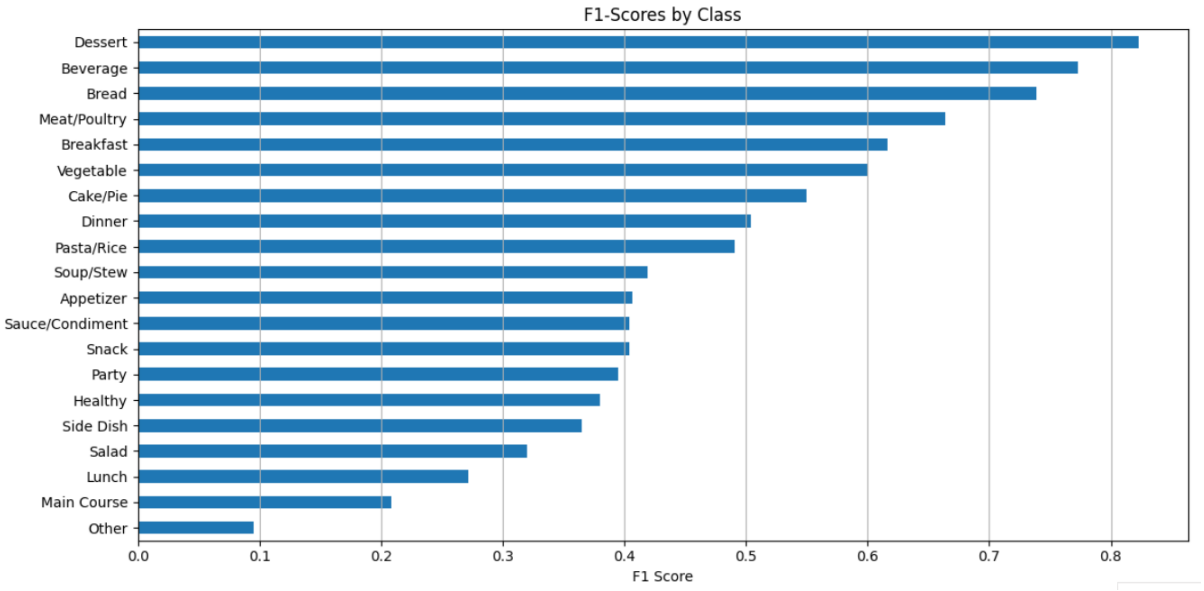


Figure 4.1: Categories F1 SCORE

Chapter 5

Doc2Vec Model Recipe Classification

As part of our efforts to explore traditional and hybrid methods for textual recipe classification, we implemented a robust pipeline built upon the Doc2Vec algorithm for text embedding. This section outlines the theoretical foundation, implementation methodology, and evaluation of our Doc2Vec-based approach, which aimed to categorize recipes into their respective food categories using a combination of semantic vectorization and classical machine learning classifiers.

5.1 Motivation and Overview

Modern natural language processing tasks often rely on deep learning models; however, vector embedding techniques such as Word2Vec and Doc2Vec remain powerful tools in extracting latent semantic patterns from unstructured text. Unlike token-based models that encode individual words or sequences, Doc2Vec provides a dense representation for entire documents, thereby making it particularly suitable for encoding full-length recipes. Our hypothesis was that by representing each recipe—including its title, ingredients, and instructions—as a dense vector, we could capture sufficient contextual cues to distinguish between different categories of food.

In this architecture, the textual embeddings generated from Doc2Vec are combined with structured metadata (e.g., preparation time, number of servings, and region/cuisine) to form a comprehensive feature vector. These feature vectors were then passed through a series of traditional machine learning classifiers to evaluate predictive performance.

5.2 Data Preparation and Cleaning

Before vectorizing the textual content, we undertook several preprocessing steps to ensure consistency and reliability in the inputs. Each recipe contained multiple textual fields, including the recipe title, a list of ingredients, and step-by-step cooking instruc-

tions. These fields were concatenated to create a unified document per recipe. Special characters, extraneous whitespaces, and inconsistent formats were cleaned using regular expressions. Missing values across fields were uniformly replaced with placeholders such as "Unknown" to avoid information loss during modeling.

We also handled numerical fields like preparation time, servings, and calories per serving by scaling them using scikit-learn's StandardScaler to ensure compatibility with distance-based classifiers.

Additionally, the Region/Cuisine attribute, which serves as a categorical indicator of the geographical origin or style of the recipe, was one-hot encoded to allow machine learning models to interpret it as discrete inputs without introducing ordinal bias.

5.3 Text Vectorization using Doc2Vec

We employed the Doc2Vec model from the Gensim library to convert each recipe's full textual content into a fixed-size numerical embedding. Each recipe document was assigned a unique tag and passed to Gensim's TaggedDocument structure. The Doc2Vec model was trained using the distributed memory (DM) architecture, with a vector size of 50 dimensions. The model used a context window of size 5 and ignored infrequent words with a minimum count threshold of 2.

The training process ran for 20 epochs to ensure convergence. After training, we used the infervector() method to obtain vector representations for all recipes. These vectors are designed to capture the semantic relationships among ingredients, cooking styles, and procedural steps, even across syntactically diverse recipes.

5.4 Feature Concatenation and Final Representation

To enrich the vector representations, we concatenated the 50-dimensional Doc2Vec embeddings with three standardized numerical features (Preparation Time, Servings, Calories per Serving),One-hot encoded Region/Cuisine vectors. The resultant feature vector for each recipe contained both linguistic and structured dimensions, allowing the downstream classifiers to benefit from a multi-modal input space.

5.5 Classifier Architecture and Training

We experimented with several classical machine learning algorithms using the scikit-learn library.

The classifiers used include:

- Logistic Regression
- Support Vector Machine (SVM) with linear kernel
- Random Forest Classifier
- K-Nearest Neighbors (KNN) with k=5
- Multinomial Naive Bayes

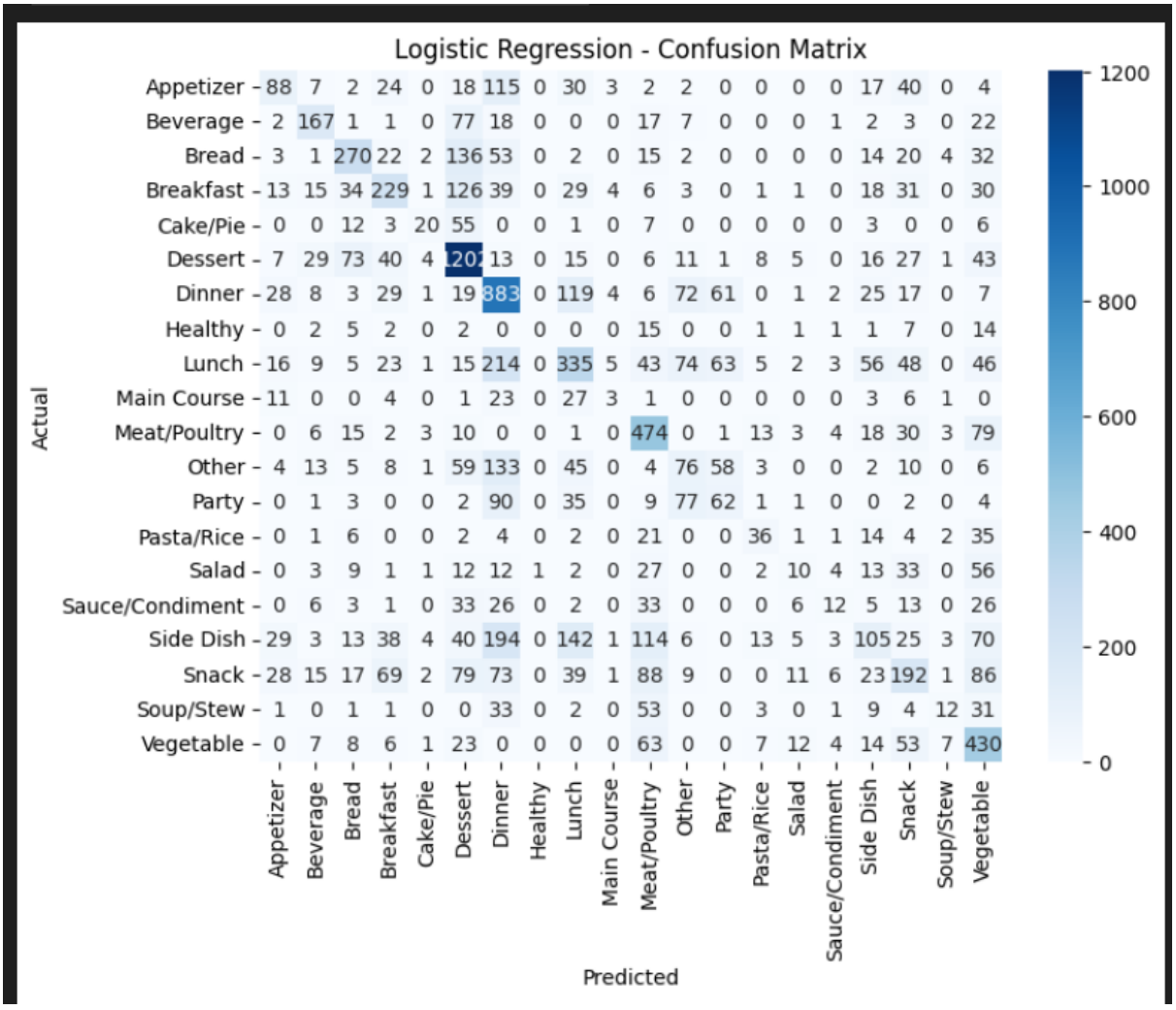
The entire dataset was divided into an 80-20 train-test split, and the models were trained using default or slightly tuned hyperparameters. We evaluated these models using accuracy, F1-score, and the confusion matrix to measure performance.

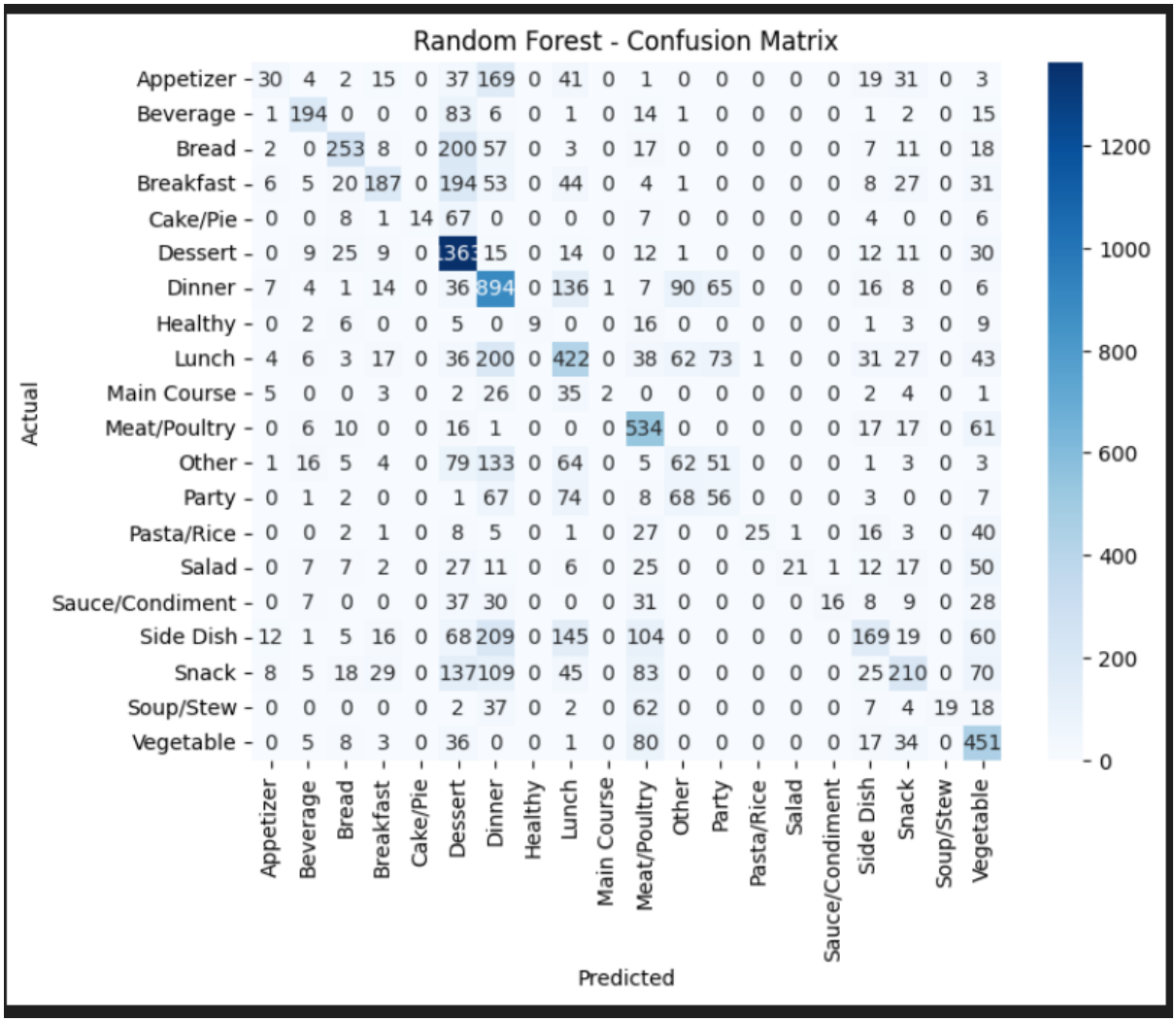
5.6 Evaluation, Results And Conclusion

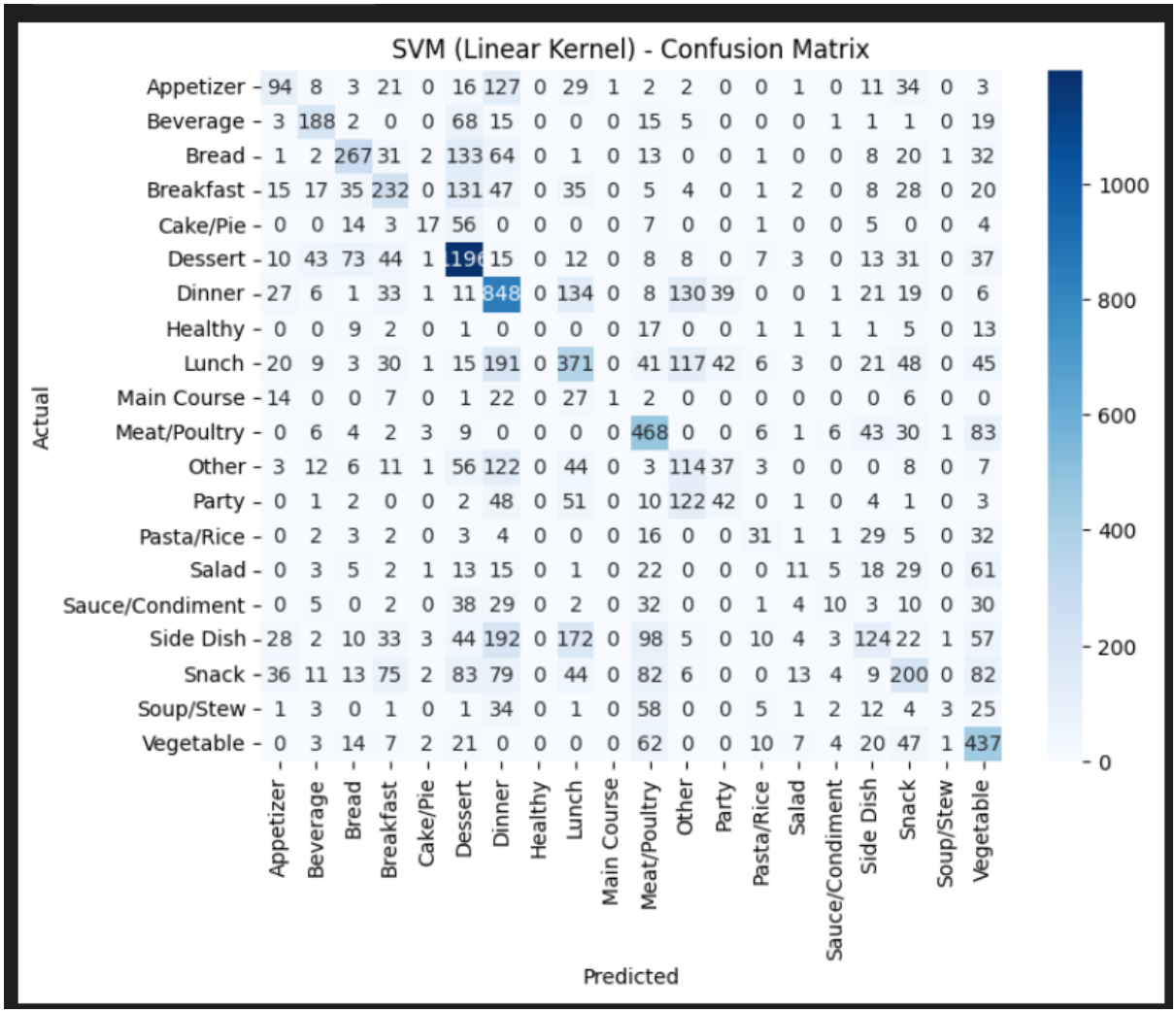
The evaluation showed that both Logistic Regression and Random Forest performed consistently well on the test set. The Random Forest classifier achieved the highest accuracy, followed closely by Logistic Regression. KNN and Naive Bayes showed lower performance, which we attribute to their limited capacity to model complex high-dimensional relationships, especially in sparse or one-hot encoded inputs.

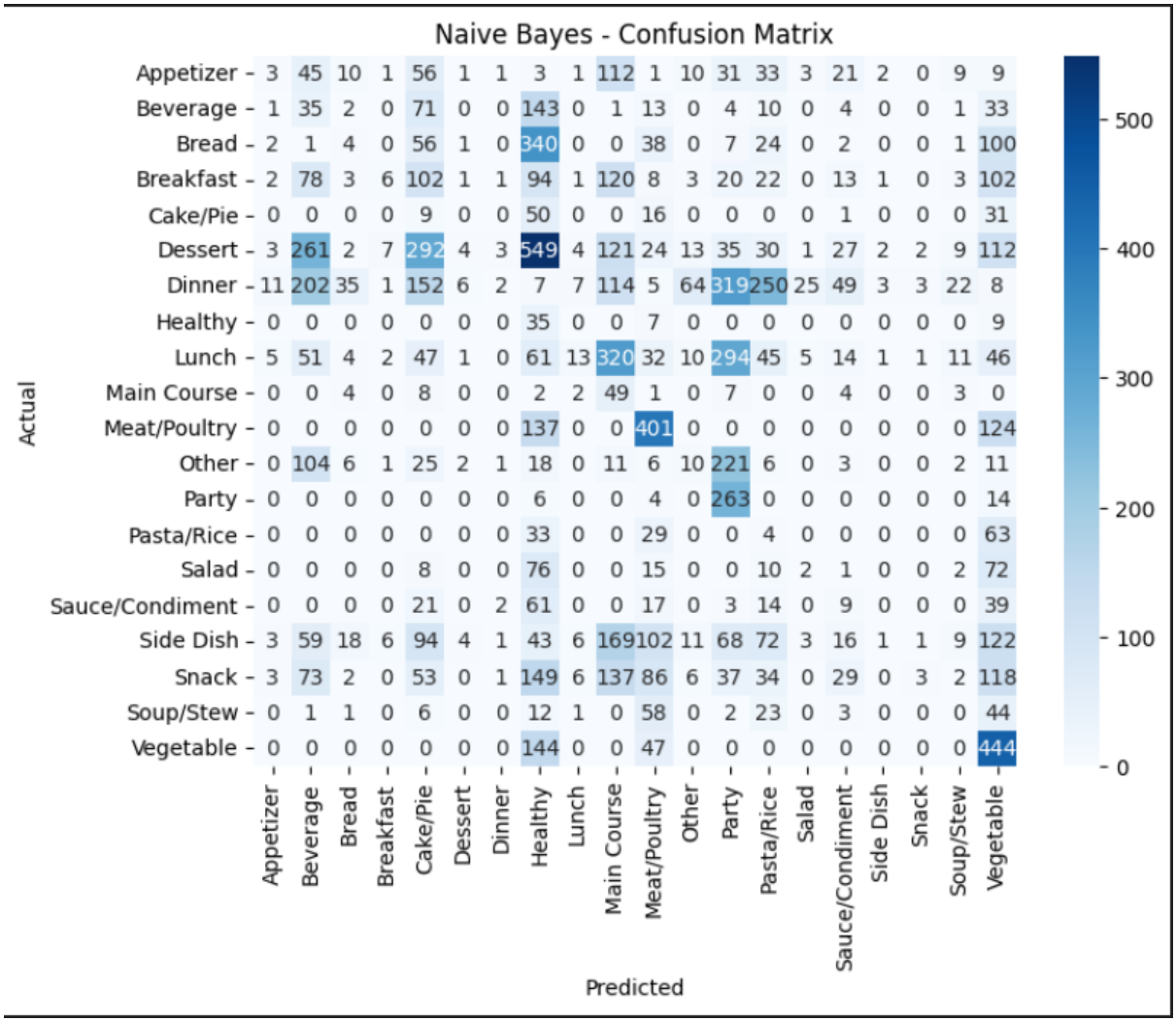
The confusion matrices revealed that certain food categories (e.g., Dessert, Appetizer) were more easily separable, while others such as "Side Dish" and "Main Course" had more overlap. This indicated that some categories might be semantically close in terms of ingredients and cooking techniques, which remains a challenge for any classification model.

The results also demonstrated the strength of combining dense embeddings with structured features. Recipes that may not have had obvious linguistic cues were still correctly classified due to supporting metadata such as cuisine or preparation time.


Logistic Regression - Confusion Matrix
Actual
Predicted
Appetizer
Beverage
Bread
Breakfast
Cake/Pie
Dessert
Dinner
Healthy
Lunch
Main Course
Meat/Poultry
Other
Party
Pasta/Rice
Salad
Sauce/Condiment
Side Dish
Snack
Soup/Stew
Vegetable
1200
1000
800
600
400
200
0


Random Forest - Confusion Matrix
Actual
Predicted
Appetizer
Beverage
Bread
Breakfast
Cake/Pie
Dessert
Dinner
Healthy
Lunch
Main Course
Meat/Poultry
Other
Party
Pasta/Rice
Salad
Sauce/Condiment
Side Dish
Snack
Soup/Stew
Vegetable
1200
1000
800
600
400
200
0


SVM (Linear Kernel) - Confusion Matrix
Actual
Predicted
Appetizer 94 8 3 21 0 16 127 0 29 1 2 2 0 0 1 0 11 34 0 3
Beverage 3 188 2 0 0 68 15 0 0 0 15 5 0 0 0 1 1 1 0 19
Bread 1 2 267 31 2 133 64 0 1 0 13 0 0 1 0 0 8 20 1 32
Breakfast 15 17 35 232 0 131 47 0 35 0 5 4 0 1 2 0 8 28 0 20
Cake/Pie 0 0 14 3 17 56 0 0 0 0 7 0 0 1 0 0 5 0 0 4
Dessert 10 43 73 44 1 196 15 0 12 0 8 8 0 7 3 0 13 31 0 37
Dinner 27 6 1 33 1 11 848 0 134 0 8 130 39 0 0 1 21 19 0 6
Healthy 0 0 9 2 0 1 0 0 0 0 17 0 0 1 1 1 1 5 0 13
Lunch 20 9 3 30 1 15 191 0 371 0 41 117 42 6 3 0 21 48 0 45
Main Course 14 0 0 7 0 1 22 0 27 1 2 0 0 0 0 0 0 6 0 0
Meat/Poultry 0 6 4 2 3 9 0 0 0 0 468 0 0 6 1 6 43 30 1 83
Other 3 12 6 11 1 56 122 0 44 0 3 114 37 3 0 0 0 8 0 7
Party 0 1 2 0 0 2 48 0 51 0 10 122 42 0 1 0 4 1 0 3
Pasta/Rice 0 2 3 2 0 3 4 0 0 0 16 0 0 31 1 1 29 5 0 32
Salad 0 3 5 2 1 13 15 0 1 0 22 0 0 0 11 5 18 29 0 61
Sauce/Condiment 0 5 0 2 0 38 29 0 2 0 32 0 0 1 4 10 3 10 0 30
Side Dish 28 2 10 33 3 44 192 0 172 0 98 5 0 10 4 3 124 22 1 57
Snack 36 11 13 75 2 83 79 0 44 0 82 6 0 0 13 4 9 200 0 82
Soup/Stew 1 3 0 1 0 1 34 0 1 0 58 0 0 5 1 2 12 4 3 25
Vegetable 0 3 14 7 2 21 0 0 0 0 62 0 0 10 7 4 20 47 1 437
Appetizer
Beverage
Bread
Breakfast
Cake/Pie
Dessert
Dinner
Healthy
Lunch
Main Course
Meat/Poultry
Other
Party
Pasta/Rice
Salad
Sauce/Condiment
Side Dish
Snack
Soup/Stew
Vegetable
1000
800
600
400
200
0



This Doc2Vec-based pipeline provided a strong and interpretable baseline for recipe classification. It combined the semantic expressiveness of document embeddings with the efficiency and transparency of traditional machine learning models. Although this method was outperformed by more complex deep learning architectures in later experiments (e.g., BiGRU + CNN with attention), its simplicity, computational efficiency, and respectable performance make it a viable candidate for lightweight inference in real-world applications.

Chapter 6

Fine-tuned Model: RecipeBERT

To address the limitations of the baseline, a fine-tuned domain-specific model, RecipeBERT, was introduced. RecipeBERT is pre-trained on a large corpus of cooking-related texts and understands the structure and semantics of recipes. The model was fine-tuned on the recipe classification dataset to better capture the differences between categories such as snacks, beverages, or desserts. This approach aimed to significantly enhance performance while maintaining the same classification pipeline structure.

6.1 Text Vectorization Using RecipeBERT

Input texts (name, ingredients, instructions) were tokenized using the RecipeBERT tokenizer. The model was fine-tuned, allowing it to adjust internal weights to better fit the recipe classification task. Like BERT, the [CLS] token embedding from the final hidden state was used as the feature representation for classification. The Hugging Face Trainer API was used to handle the training pipeline, tokenization, and data loading efficiently.

6.2 Feature Concatenation and Final Representation

The recipe text was treated as a single sequence, and the model processed the full sequence jointly. The [CLS] representation encoded context from all parts of the input (ingredients, instructions, etc.) and was directly used for classification.

6.3 Classifier Architecture and Training

The RecipeBERT model included a classification head: a linear layer stacked on top of the transformer encoder. Training Details: Epochs: 9 Batch Size: 8 Weight Decay: 0.01 Early Stopping: Patience set to 2 Best model checkpoint saved based on validation accuracy.

6.4 Evaluation, Results And Conclusion

Test Accuracy: 59.98 percentage Loss Trend: Both training and validation loss steadily declined across all epochs. Observations: No signs of overfitting or instability. The domain alignment of RecipeBERT with culinary data enabled improved classification. Performance was the best among all models tested in the project.

Fine-tuning RecipeBERT for the classification task showed a substantial improvement in model performance, nearly doubling the accuracy over the baseline. The model's specialization in recipe-related language allowed it to better understand subtle category differences. This demonstrates the power of domain-specific pre-training and fine-tuning in NLP tasks, and establishes RecipeBERT as the most effective configuration in the current phase of the BTP project.

Chapter 7

Conclusion

In this project, we have demonstrated a comprehensive end-to-end pipeline for applying Natural Language Processing techniques to recipe texts, with the goal of accurately classifying recipes into standardized meal categories. Beginning with the acquisition of a large, diverse corpus from The Spruce Eats, Archana's Kitchen, and Food.com, we developed robust data-cleaning and normalization procedures that reduced noise, unified formats, and consolidated over 600 initial labels into 20 semantically coherent categories. Our exploratory data analysis revealed key insights—such as ingredient-richness differences between desserts and beverages, and distinct preparation-time distributions—that informed subsequent modeling decisions and highlighted the value of combining textual and structured metadata.

We evaluated a spectrum of modeling approaches. Traditional embedding methods (Doc2Vec) paired with classical classifiers (Logistic Regression, SVM, Random Forest, KNN, Naive Bayes) achieved respectable baseline performance, illustrating the predictive power of dense vector representations coupled with simple algorithms. Building on this, our deep learning architectures—a vanilla CNN-BiGRU model and a hybrid BiGRU-CNN with multi-head attention and auxiliary features—demonstrated meaningful gains by capturing both local n-gram patterns and long-range dependencies in instruction text. In particular, incorporating instruction length and attention mechanisms yielded clearer separation between semantically similar categories, raising validation accuracy by roughly 4 %. The most significant improvement came from transformer-based methods. A frozen BERT-embedding pipeline served as a stepping stone, but the fine-tuned RecipeBERT model—pre-trained on culinary corpora—proved superior, boosting classification accuracy to nearly 60 %. This underscores the importance of domain-specific language models in handling specialized vocabulary and structural idiosyncrasies inherent in recipe texts. Despite these advances, several challenges remain. Class imbalance still affects performance on underrepresented categories, and pure text-based classification cannot leverage visual cues from recipe images or ingredient measurements. Moreover, our current system treats each recipe independently; future work could explore multi-task frameworks

that jointly predict nutritional information or cooking difficulty. Finally, deployment considerations—such as inference latency, memory footprint, and integration into real-time cooking assistants—will inform the choice between lightweight Doc2Vec-based systems and heavier transformer solutions.

7.1 : Key Takeaways Future Directions

- Data Quality Matters: Rigorous cleaning, deduplication, and category consolidation set the stage for reliable model training.
- Model Spectrum: From classical ML to deep hybrids to transformers, each method offers trade-offs in complexity, interpretability, and accuracy.
- Domain Adaptation: Fine-tuning RecipeBERT was critical for capturing cooking-specific semantics.
- Multi-Modal Extensions: Incorporating images, ingredient quantities, and user ratings could further boost classification.
- Real-World Deployment: Next steps include packaging models as API services, optimizing for edge devices, and conducting user studies within meal-planning applications.

In summary, our work establishes a solid foundation for NLP-driven recipe understanding, balancing methodological rigor with practical considerations. Building on these results, future research can expand into recommendation systems, nutritional analysis, and multilingual recipe corpora—ultimately empowering smarter culinary tools and personalized cooking experiences.

Bibliography

- Le, Q., Mikolov, T. (2014). Distributed Representations of Sentences and Documents. Proceedings of ICML.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781.
- Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. EMNLP, 1746–1751.
- Cho, K., van Merriënboer, B., Gulcehre, C., et al. (2014). Learning Phrase Representations Using RNN Encoder–Decoder for Statistical Machine Translation. EMNLP, 1724–1734.
- Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention Is All You Need. NeurIPS, 5998–6008.
- Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL-HLT, 4171–4186.
- Seo, A. (2024). RecipeBERT: A Transformer Model for Culinary Language Understanding [GitHub]. Available at: https://huggingface.co/alexdseo/RecipeBERT
- Pedregosa, F., Varoquaux, G., Gramfort, A., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825–2830.
- Richardson, L. (2007). Beautiful Soup Documentation. Available at: https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Bird, S., Klein, E., Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media.
- The Spruce Eats. (Accessed April 23, 2025). Recipes. Available at: https://www.thespruceeats.com/

- Archana's Kitchen. (Accessed April 23, 2025). Recipes. Available at: https://www.archanaskitchen.com/
- Food.com. (Accessed April 23, 2025). Recipes. Available at: https://www.food.com/