Chef Classification Task - NLP Project

João Lopes, Kun Fang, Tomás Henriques, and Tomás Cruz

Group 64
Instituto Superior Técnico, Universidade de Lisboa

October 15, 2025

1 Introduction

The task addressed in this project is a multi-class classification problem. The goal is to assign each recipe to one of six chefs, using features extracted from each recipe's name, tags, steps, description and ingredients. The dataset contains a total of 2,999 recipes with moderate class imbalance (Table 1). The challenge lies in distinguishing recipes that may share similar ingredients or steps, requiring models that can capture subtle textual differences.

2 Models

Two main approaches were explored: a recurrent neural network (RNN) and a linear support vector classifier (LinearSVC).

2.1 Preprocessing

For LinearSVC, all fields were concatenated, lowercased, tokenized and transformed into TF-IDF features using unigrams and bigrams. Stopwords were also removed to reduce noise. For the RNN, fields were concatenated and tokenized, however, without any removal of stopwords, to preserve sequential information. No data augmentation was applied.

2.2 Model Architecture

The RNN model is a bidirectional GRU-based Recurrent Neural Network (RNN) designed to classify recipes by chef. Recipe text fields are concatenated, tokenized, and converted into numerical sequences using a limited vocabulary with `<PAD>` and `<UNK>` tokens. An embedding layer maps tokens to dense vectors, which are processed by the GRU to capture contextual patterns in both directions. An attention layer aggregates the most informative features before passing them through a fully connected classifier. Training uses cross-entropy loss with class weighting, the AdamW optimizer, and automatic mixed precision to reduce GPU usage. Early stopping and learning rate scheduling prevent overfitting, while dynamic padding via a custom `DataLoader` ensures efficient batching and memory usage. LinearSVC was used with optimized hyperparameters obtained from a GridSearchCV on a subset of the training data.

3 Experimental Setup

3.1 Datasets

The original dataset was split into training and testing sets using four different ratios: 0.5, 0.7, 0.8, and 0.9. For each split, the dataset was shuffled to avoid repeating the same training/testing

distributions.

3.2 Evaluation Metrics

Accuracy, macro F1-score and weighted F1-score were used to evaluate model performance. Confusion matrices were also generated to assess per-class performance and to analyze common misclassifications.

3.3 Hyperparameters

The best hyperparameters for the RNN were: hidden_dim = 128, embed_dim = 128, dropout = 0.3, training epochs = 30 and MAX_LEN = 100. LinearSVC used: C = 2.0, TF-IDF max_df = 1.0, min_df = 2, ngram_range = (1, 2).

4 Results

Table 2 summarizes the accuracy and F1-scores for RNN and LinearSVC across different splits. LinearSVC consistently outperformed the RNN, reaching up to 95.33% accuracy on the 0.9 split, while RNN peaked at 77.96%.

4.1 Confusion Matrix Analysis

Figure 1 shows the confusion matrix for LinearSVC with split 0.8. Most misclassifications occur for chefs with fewer samples, such as 1533 and 6357, reflecting the moderate imbalance in the dataset. The RNN confusion matrices show more distributed errors across classes, indicating difficulty in learning distinguishing features from the recipes' sequential information.

5 Discussion

The LinearSVC consistently outperformed the RNN across all splits. This result can be attributed to the nature of the dataset: distinguishing features are largely captured by word occurrence and frequency rather than sequence, favoring bag-of-words approaches. The RNN struggled, likely due to insufficient data to learn meaningful sequential patterns for each of the six classes.

Errors identified in the confusion matrices mainly involve chefs with fewer recipes (1533 and 6357), consistent with the moderate class imbalance (13–27%). The RNN frequently misclassified recipes between chefs with overlapping vocabulary in ingredients and steps, while LinearSVC effectively separated classes using TF-IDF weighted word features.

6 Future Work

Further improvements could include:

- Applying data augmentation techniques such as synonym replacement or recipe paraphrasing to increase training data diversity and balance.
- Experimenting with pre-trained embeddings or transformer-based models, which may better capture contextual relationships in text.
- Implementing class weighting or oversampling to address moderate imbalance.

References

[1] GeeksforGeeks. *RNN for Sequence Labeling*. https://www.geeksforgeeks.org/deep-learning/rnn-for-sequence-labeling/

[2] Python Tutorials. *Introduction to the Use of LinearSVC*. https://www.pythontutorials.net/blog/sklearn-linearsvc/

[3] Course Material. *Sebenta, Part 1 and 2 (available on the course page)*.

[4] Stanford NLP Group. *GloVe: Global Vectors for Word Representation.* https://github.com/stanfordnlp/GloVe

Appendix A: Additional Figures and Tables

Table 1: Percentage of Recipes per Chef

Chef ID	Count	Percentage (%)
1533	404	13.47
3288	451	15.04
4470	806	26.88
5060	534	17.81
6357	372	12.40
8688	432	14.40

Table 2: Model Performance by Split Ratio

Split	Model	Accuracy	F1-macro
0.5	RNN	0.7433	0.7128
0.5	LinearSVC	0.9207	0.9108
0.7	RNN	0.7762	0.7623
0.7	LinearSVC	0.9367	0.9283
0.8	RNN	0.7646	0.7508
0.8	LinearSVC	0.9400	0.9311
0.9	RNN	0.7796	0.7682
0.9	LinearSVC	0.9533	0.9469

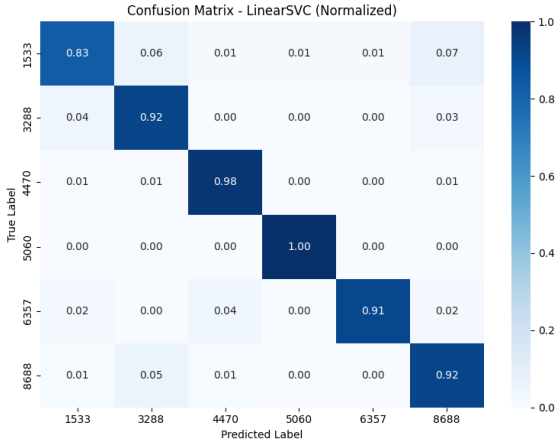

Figure 1: Confusion matrix of LinearSVC on split 0.8. Rows correspond to true labels, columns to predicted labels.

Acknowledgements

For transparency purposes, we further add that AI was used to generate code snippets and to correct this report grammatically.