Recipe Generation System

Harshitha Biligere Harish (hbilige)
Tanvi Kulkarni (tanakulk)
Amit Singh (amisingh)
Kalyani Malokar (kmalokar) [1*]

Abstract
The goal of this project is to develop a system that can generate a recipe and detect the cuisine based on a list of ingredients provided by the user. Oftentimes people are faced with the problem of what to cook with the limited ingredients that are available. It is not always possible to come up with new recipes, explore different cuisines and eating the same repetitive dishes can get monotonous after a point. Getting a recipe recommendation and understanding the cuisine can certainly make things easier and helps to save cooking time as well. We will use Natural Language Processing (NLP) techniques to analyze the list of ingredients and generate a recipe using RNN as well as conduct multiclass classification to detect the cuisine. This system will not only provide a practical solution for individuals who are short on time or seeking inspiration in the kitchen, but also serve as a foundation for future research and development in the culinary field.

Keywords
Recurrent Neural Networks (RNN) — Natural Language Processing(NLP) — Recipe Generation — Cuisine detection — TensorFlow — Multiclass classification

[1] *Luddy School of Informatics, Computing, and Engineering, Indiana University, Bloomington, IN, USA*

Contents

1. Problem and Data Description

1.1 Data Mining problem:

In recent times, people have become increasingly health-conscious and prioritize consuming nutritious food. A good food recipe should both enhance the flavor of the food and provide a balanced diet for a healthy lifestyle. Since good health is valuable, selecting the right ingredients is crucial.

Choosing healthy ingredients to cook with is pretty straightforward. However, beginners as well as expert cooks may choose ingredients without knowing how to use them in a recipe. Also, there are various cuisines that have their own recipes and methods of cooking with the same ingredients. In order to cook a delicious recipe, it is important to understand which ingredients complement each other and can create delicious dishes. This can be a challenging process for both novice and experienced cooks at times to select a recipe based solely on the ingredients. Deep learning and machine learning techniques can be used to overcome these issues. We want to use deep learning and machine learning for recipe generation and cuisine detection based on a list of ingredients.

1.2 Data Description:

Recipe Generation: Recipe generation involves using a curated list of ingredients along with step-by-step instructions for creating a dish. Our model for milestone 1 is built using the 'RecipeBox' dataset, which includes over 125,000 recipes scraped from three popular recipe sources - FoodNetwork, Epicurious, and AllRecipes. Due to computational limitations, we are using only the AllRecipes dataset for our current model, which includes 39,802 recipes with four attributes: title, ingredients, instructions, and picture link. The RecipeBox dataset includes recipes from various cuisines, courses, and flavor profiles, making it a diverse and comprehensive resource for recipe generation. Some examples of the recipe titles in the dataset include "Brown Sugar Meatloaf," "Homemade Mac and Cheese Casserole," "Best Chocolate Chip Cookies,".

Cuisine Detection: For cuisine detection, we have used

a dataset from Yummly. The dataset has 39774 rows and 3 attributes - ID, Cuisine and Ingredients. The dataset comprises 20 types of cuisines such as Greek, Italian, Jamaican, Mexican, Indian, etc. There are 6714 ingredients present in the dataset.

2. Data Preprocessing & Exploratory Data Analysis

2.1 Handling Missing Values

Recipe generation:

- Out of the 4 columns, picture_link will not be used for our model and hence it's dropped.
- There are 840 missing values across 3 columns. Since these are small numbers, dropping them will not affect our model by a lot. So, all NA and missing values are dropped, bringing the total number of entries to 39522.
- A histogram is plotted for length of instructions and after analyzing the plot, it can be seen that there are some outliers (after 3000 length) and noise that can be removed. After 1500 there are very few recipes so those are dropped, bringing the total recipes to 38859.

Cuisine Detection:

The dataset used for cuisine detection does not have any missing values.

2.2 Data preprocessing

Recipe generation:

- Data was converted to lower alphabetical format to maintain consistency.
- 'Title' , 'ingredients' and 'instructions' are joined into a single string and created as a column 'recipes' in a new dataframe for later usage in RNN.
- Ingredients list have unwanted word 'ADVERTISEMENT' is removed.
- New line character is replaced with space.
- Tokenization: A vocabulary of recipe objects as an array of characters is created by tokenizing using the tensorflow.keras library.
- Vectorization: RNN takes numerical values as input so all the tokens generated are converted to unique numerical values.
- Padding: All input needs to be of same length for training, so the max length (1500) is considered and all the recipe strings are post-padded with stop word to maintain the same length.

Cuisine Detection:

- Converting to lowercase: We converted all text to lower case to help in reducing the number of unique words in the dataset and to help in avoiding redundant features.
- Removing punctuation: Punctuation was removed to clean the text data and make it more consistent.
- Lemmatization: Lemmatization was done to reduce redundancy in the dataset and maintain a list of ingredients in their base/root form. This helps to reduce the number of unique words in the dataset and makes the feature space more informative.
- TF-IDF: TF-IDF was used to reduce the dimensionality, and accurately weigh the importance of each feature (i.e., ingredient) in the cuisine detection task.

2.3 Exploratory Data Analysis

Recipe generation:

Based on the analysis we performed, we can say the following about the data:

- Some of the most used ingredients are: salt, sugar, butter, pepper, onion, olive oil.
- During EDA, we discovered that we cannot remove a lot of the data as it is integral to generating the right instructions for recipe generation.
- A number of ingredients v/s instruction length is plotted and it can be seen that as the number of ingredients increases, the instruction lengths also increase indicating that it is directly dependent.
- It can also be observed that most recipes have around 20-50 ingredients with instruction lengths up to 250.
- A pie chart of top 10 ingredients is created which shows that salt and pepper are the most used ingredients in all recipes, followed by butter, garlic.
- Overall, the dataset provides a good starting point for exploring the relationship between different recipe instructions and their corresponding ingredients.

Figure 1. Word cloud for ingredients

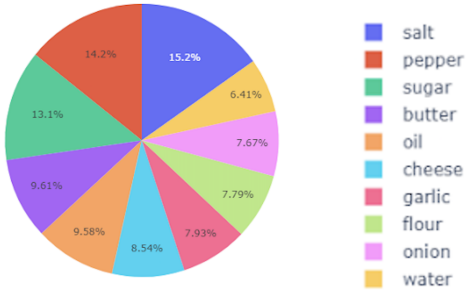

Figure 2. Pie chart for top 10 ingredients

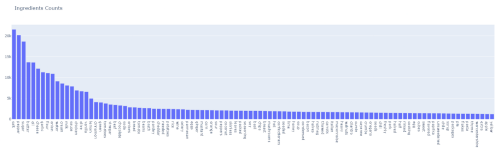

Figure 3. Ingredient counts

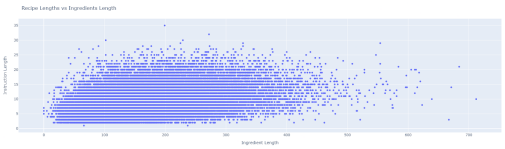

Figure 4. Ingredient counts vs instructions length

Cuisine Detection:

Based on the analysis we performed, we can say the following about the data:

- The dataset contains recipes from 20 different cuisines, with the top 5 being Italian, Mexican, Southern US, Indian, and Chinese.
- Italian recipes have the highest number of recipes in the dataset, followed by Mexican and Southern US recipes.
- The average number of ingredients per recipe varies across cuisines, with Southern US recipes having the highest average number of ingredients and Japanese recipes having the lowest.
- The most common ingredients across the top 5 cuisines are garlic, onions, olive oil, salt, and water.
- During our EDA, we discovered that removing the most common ingredients from the dataset did not provide much meaningful insight. As a result, for this analysis, we have decided to keep the most common ingredients that are present across all cuisines.
- It's important to keep in mind that the most common ingredients are often used across many different cuisines and can provide valuable insights into what makes certain ingredients so popular and versatile in cooking. Additionally, keeping the most common ingredients can also help to identify key ingredients that define a particular cuisine or region.
- There was an issue of similar ingredients being written differently, this is a common problem in natural language processing and can be addressed through techniques such as stemming, lemmatization, and entity recognition. In this case, we used some basic preprocessing techniques to standardize the ingredient names. Which has been discussed in the preprocessing section.
- Overall, the dataset provides a good starting point for exploring the relationship between different cuisines and their corresponding ingredients. We can use machine learning techniques to predict the cuisine of a dish based on its ingredients.

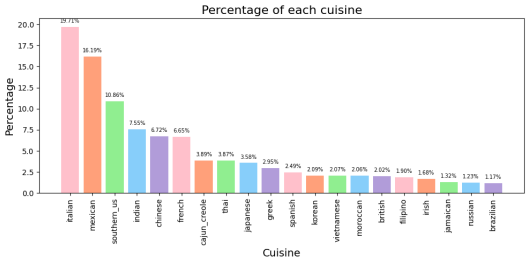

Figure 5. Percentage of cuisines

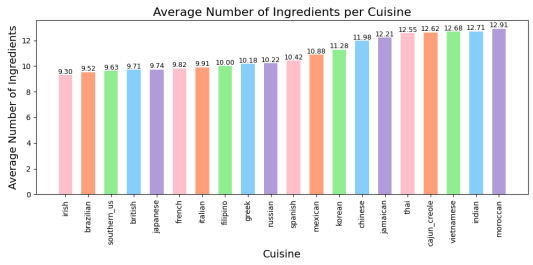

Figure 6. Average no. of ingredients per cuisine

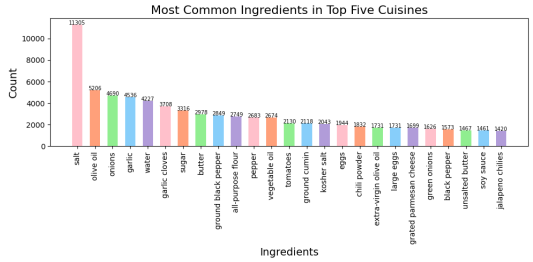

Figure 7. Most common ingredients in top 5 cuisines

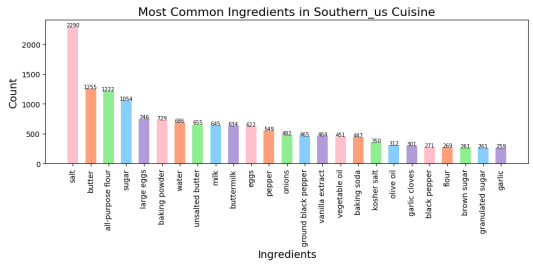

Figure 8. Most common ingredients in Southern US cuisine

3. Algorithms & Methodology

3.1 Recipe Generation

In this project, we aimed to generate recipes using various RNN models such as LSTM, GRU, and GANs. After careful evaluation, we discovered that LSTM yielded the best results. To train our model, we scraped data from three websites in JSON format and performed basic preprocessing to prepare it for an RNN model. We combined the recipe title, ingredients, and instructions to run it on RNN. Since recipes vary in length, we plotted the length of instructions vs the length of ingredients and then determined a threshold value, and hard-coded the recipe length to 2000 words to cover most recipes. Additionally, we added a `START_CHAR` and `STOP_CHAR` to indicate the start and end of each recipe. To tokenize the recipes on a character level, we employed Tensorflow's preprocessing tokenizer, which generated a vocabulary that we then vectorized. We added padding using TensorFlow's `pad_sequence` library. Lastly, we converted the dataset into a Tensorflow dataset, ready for training our models.

The Long Short-Term Memory (LSTM) architecture uses memory cells to capture long-term dependencies in sequence models. These memory cells are updated using regulatory gates, enabling the addition and deletion of information as required. In our model, we created an embedding layer with the vocabulary size as input dimensions and an output dimension of 256. The LSTM layer consisted of a single layer with 1024 units. To calculate the loss function, we employed Tensorflow's `sparse_categorical_crossentropy` library and an `Adam` optimizer with a learning rate of 0.001. We also established a directory for checkpoints to store the training weights and set up a callback function to load the model history each time it runs (provided the environment is not reset).

To generate recipes, we created a function that can produce recipes with multiple fuzziness levels ranging from 0.2 to 1.0, with increments of 0.2. Higher fuzziness levels result in more surprising text, while lower fuzziness levels result in more predictable text. We observed that for some ingredients/titles, a high fuzziness level produced better output, which could be due to the lower number of epochs used. Overall, our approach using LSTM and various techniques for dataset preparation and model creation provides promising results for generating recipes.

3.2 Cuisine Detection

In our project for cuisine detection, we aimed to classify different cuisines based on their ingredients. We explored three different classification algorithms - logistic regression, random forest, and support vector machine (SVM) to perform multiclass classification. We then trained each algorithm on the training set and evaluated their performance on the testing set to determine which algorithm gave the best results.

Logistic regression is a linear classification algorithm that uses a logistic function to model the probability of a binary target variable. It is a simple and efficient algorithm that works well for linearly separable data. We have trained our logistic regression model with a penalty term as 'l2 norm' with 'lbfgs' as the algorithm solver with `max_iter` as 100 for the solver to converge.

Random forest is an ensemble algorithm that combines multiple decision trees to make a final prediction. It works well for both categorical and continuous variables and is less prone to overfitting. We have trained our random forest model with the number of trees (`n_estimators`) as 100, quality of split is measured using gini entropy and number of features when looking for the best split has been taken as `max_features=sqrt(n_features)`.

Support vector machine (SVM) is a powerful classification algorithm that works well for both linearly separable and

non-linearly separable data. It tries to find the best hyperplane that separates the data points into different classes. We have trained our SVM model with 'rbf' kernel, regularization parameter value as '1.0' and gamma has been set to 'scale'.

4. Experiments & Results

4.1 Recipe Generation

To evaluate the recipe generation model, we provided the model with different ingredients and titles, varying the fuzziness levels to achieve a range of results. Higher fuzziness levels were associated with more surprising outcomes, while lower fuzziness levels resulted in more predictable text. We tested both LSTM and GRU models and found that the GRU model incurred more loss and produced less accurate recipes than the LSTM model.

The model was further evaluated on multiple ingredient combinations, including those that made sense together and those that did not. In most cases, the model generated legible recipe instructions, with specific instructions for plausible ingredient lists or recipe titles. Even for random ingredient combinations, the model was able to generate a recipe but lacked coherence. The model also occasionally removed or added extra ingredients to create a viable recipe. Overall, the recipe generation model demonstrated good results, though it could have performed better with more epochs or better computation resources.

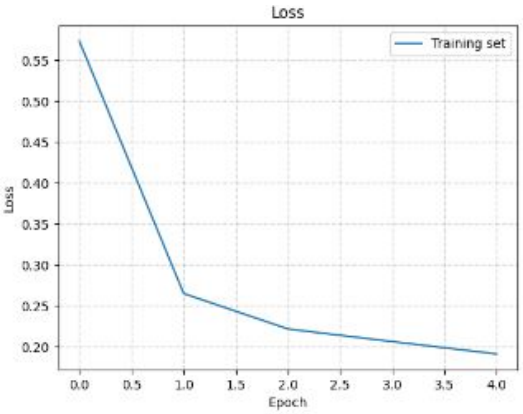

Figure 9. Loss function for LSTM

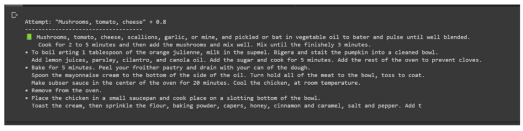

Figure 10. Sample

4.2 Cuisine Detection

We have used the following metrics to evaluate the performance of the 3 models built for cuisine detection:

- Accuracy: Provides a simple and intuitive measure of how well the algorithm is performing. It is the proportion of correctly classified instances out of the total number of instances in the dataset.
- Precision: Precision is defined as the ratio of True Positive (correctly classified positive samples) to the total number of samples classified as positive.
- Recall: Recall is defined as the ratio of True Positive (correctly classified positive samples) to the total number of positive samples.
- F1 Score: F1 score is the harmonic mean of precision and recall, i.e., it combines them into a single metric. It works well on imbalanced data.

Models	Accuracy Score	Macro average scores		
		Precision	Recall	F1 score
Logistic Regression	78.13	74	67	70
SVM	79.39	80	68	72
Random Forest	71.95	75	54	60

Figure 11. Model performance comparison

The SVM model has the highest accuracy score (79.39). The Precision score of SVM (80) is also higher than the other two models, indicating that it has a higher percentage of correct positive predictions. However, the Recall score of SVM (68) is lower than the Precision score, which means that it may be missing some true positive instances.

The Logistic Regression model has a lower accuracy score (78.13) than the SVM model. The Random Forest model has the lowest performance in terms of both Recall (54) and F1 score (60), indicating that it may be missing a significant number of true positive instances.

In summary, the SVM model has the highest accuracy, precision and f1 scores, while Logistic Regression has a higher recall score (67) compared to the SVM model (68), which is closer to its precision score (74). Since the specific task requires cuisines to be predicted accurately based on above evaluation, SVM proved to be the most effective algorithm for our cuisine detection part of the project.

5. Deployment & Maintenance

To deploy the app locally, we used Python's Streamlit library. After creating the app file, we used the "streamlit run" command followed by the file name to launch the app in a web browser. The app was then accessible locally at http://localhost:8501.

To make the app available to other users on the local network,

we shared the URL along with the port number. To ensure the app remains functional and up-to-date, we will maintain it by keeping the dependencies and libraries used in the code up-to-date. We will also monitor the app's performance and usage to identify and fix any issues that may arise. Automated testing and continuous integration will be set up to prevent any changes to the code from breaking the app's functionality. Overall, deploying the app locally using Python's Streamlit library provides an accessible and easily maintainable solution. By following best practices for app maintenance, we can ensure that the app remains functional and useful over time.

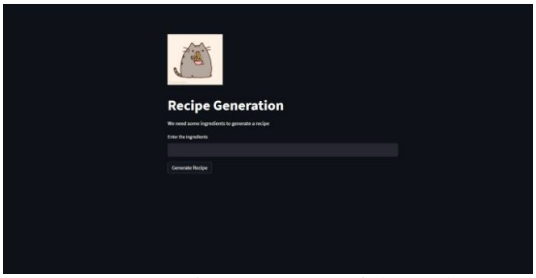

Figure 12. Static Front Page of the WebApp

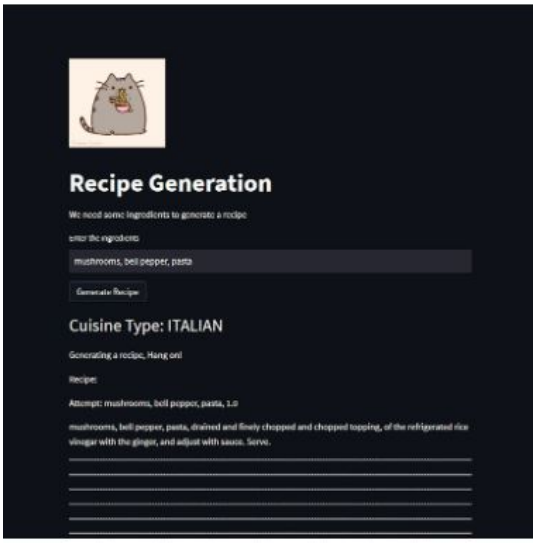

Figure 13. Classifying Recipe type and Generating Recipe (1)

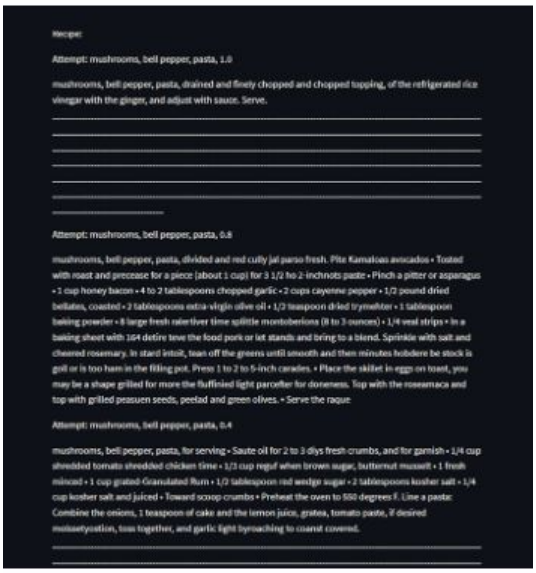

Figure 14. Generating Recipe (2)

Figure 15. Generating Recipe (3)

6. Summary & Conclusions

In our Recipe Generation and Cuisine Detection system, we aimed to experiment with multiple models and select the one that produces the best results. We succeeded in doing so, as most models we came across were limited to either recipe generation or cuisine detection, lacking the necessary attributes and multiple classifications. Initially, our aim was to generate a recipe based on a set of ingredients and detect the cuisine for the generated recipe. However, we were not able to get data which collectively included ingredients, recipe, and cuisine in

order to cohesively link both the functionalities of recipe generation and cuisine detection. For Recipe Generation, we used scraped data consisting of recipe titles, instructions and ingredients to train a LSTM model, while another dataset consisting of ingredients and cuisine was used to build SVM model for Cuisine Detection. Our implementation achieved similar accuracy and scores as other implementations of the problem, with better results attainable through multiple epochs of training or increased computational resources. Although not perfect, our model generated outputs in a human-readable format and aligned with expected results. As part of future work, we plan to include additional attributes such as nutrition value, time to cook, and flavor profiles. And explore different models such as transformers, GANs, and multi-layer LSTMs. Currently, our model detects cuisine based on the ingredients used, but we would like to extend this to detect cuisine based on specific recipe instructions generated. Additionally, we aim to make the UI more interactive. Overall, our final implementation lines up with our initial proposal. We got insights on the working of various machine learning models and how it could be used for different tasks. It was a great learning experience and will surely help us understand natural language processing and Data Mining tasks better.

7. Acknowledgements

We would like to thank our professor, Dr. Hasan Kurban, for guiding us through this course and supporting us. We would also like to express gratitude towards the TAs for their insights and feedback throughout the semester, which helped us refine our approaches. Furthermore, we would like to thank our peers for their continued encouragement. Finally, I would like to thank my project teammates for their hard work and dedication, which was necessary for achieving our goals. Collaborating together was fun and knowledgeable.

References

[1] https://www.kdnuggets.com/2020/07/generating-cooking-recipes-using-tensorflow.html

[2] https://github.com/navassherif98/Recipe- Generation-from-Food-Image

[3] https://towardsdatascience.com/building-a-food-recommendation-system-90788f78691a

[4] https://towardsdatascience.com/using-machine-learning-to-generate-recipes-that-actually- works-b2331c85ab72

[5] https://towardsdatascience.com/generating- cooking-recipes-using-tensorflow-and-lstm-recurrent-neural-network-a7bf242acad3

[6] https://www.kaggle.com/competitions/whats- cooking/data

[7] https://eightportions.com/datasets/Recipes/

[8] https://github.com/trekhleb/machine-learning-experiments /blob/master/experiments/ // `recipe_generation_rnn/recipe_generation_rnn`.ipynb

[9] https://intellipaat.com/blog /*what − is − lstm*/: :text=First/2C/ 20you/20must/20be/ // 20wondering,especially/20in/20sequence/ 20prediction/ 20problems

[10] https://www.tensorflow.org/`api_docs`/python /tf/keras/layers/LSTM

[11] https://www.analyticsvidhya.com/blog/ 2021/03/introduction-to-gated-recurrent-unit-gru/