Generating Recipe Names using Deep Learning

Lauren Bourque
The William States Lee College of Engineering
University of North Carolina at Charlotte
Charlotte, United States of America
lbourqu1@uncc.edu

Hunter Burnett
The William States Lee College of Engineering
University of North Carolina at Charlotte
Charlotte, United States of America
hburnet7@uncc.edu

Aidan Cowan
The William States Lee College of Engineering
University of North Carolina at Charlotte
Charlotte, United States of America
acowan8@uncc.edu

***Abstract*—This paper presents a deep learning-based approach to generating recipe names from a given text prompt. The proposed model uses a sequence-to-sequence architecture with a recurrent neural network (RNN) and a vocabulary of unique characters. The model is trained on a dataset of over 13,000 recipe names from Epicurious and evaluated using metrics such as BLEU score and precision. The results show that the model can generate coherent and contextually appropriate recipe names.**

I. Introduction and Motivation

The objective of this project was to use a dataset of over 13,000 recipe names from Epicurious to train a model using sequence-to-sequence learning to generate new recipe titles when given a text prompt. Various model types such as RNN, LSTM, GRU, and Transformer-based architectures were tested and their results were compared. The potential impact includes generating new recipe ideas and using generative AI to create new recipe titles from images of foods.

The link to the GitHub repository can be found here: Click here to view the code

II. Dataset and Training Setup

The dataset consisted of 13,000+ recipe names from Epicurious. The initial project goal was to use 7,000 recipe reviews from Food.com to create personalized recipe recommendations, but this was found to be solvable using cosine similarity rather than deep learning. The Food.com dataset was found to be heavily biased to have mostly five-star reviews, which skewed the dataset. The team thought about training a model with the recipe reviews from only one user to create personalized recommendations, but the dataset didn't have a strong means for sequencing. Thus, the team redirected to using the Epicurious dataset.

III. Approach

The data preprocessing steps included dropping all columns except recipe names, removing non-English characters, creating a vocabulary list of unique characters for embedding, tokenizing the input features, and splitting the dataset into training, validation, and test sets using a 60/20/20 split. The flowchart for this process can be seen in Figure 1 below.

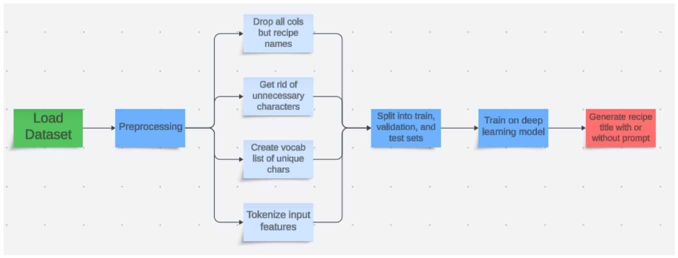

Load Dataset
Preprocessing
Drop all cols but recipe names
Get rid of unnecessary characters
Create vocab list of unique chars
Tokenize input features
Split into train, validation, and test sets
Train on deep learning model
Generate recipe title with or without prompt

Fig. 1. Pre-processing architecture.

For training, the data was processed with batch sizes of 32. The output size of the embedding layers in each model was 256. The output size of the RNN layer was 128. An Adam optimizer was chosen and Cross Entropy Loss was used. The model was trained for 20 epochs.

The model structures explored included the following architectures:

A. *Recurrent Neural Network (RNN)*

The core of the recipe name generation model is an RNN model called 'RecipeRNN'.This model implements the following key components:

a) Embedding Layer: This layer maps the input characters (represented as indices) to dense vector representations using an embedding matrix. The size of the embedding matrix is determined by the vocabulary size and the embedding dimension.

b) Recurrent Layer: The recurrent layer is a vanilla RNN that processes the embedded inputs sequentially. The RNN layer has a specified number of units, which determines the size of the hidden state.

c) Dense Layer: The output of the RNN layer is passed through a dense (fully connected) layer that maps the RNN output to the vocabulary size. This allows the model to generate the next character in the sequence.

B. *Gated Recurrent Unit Network (GRU)*

The GRU is a variant of the RNN that introduces gating mechanisms to better control the flow of information through the network. This helps the model capture long-term dependencies in the input sequence more effectively.

a) Embedding Layer: This layer maps the input characters (represented as indices) to dense vector representations using an embedding matrix. The vocabulary size and the embedding dimension determine the size of the embedding matrix.

b) GRU Layer: The recurrent layer is a GRU that processes the embedded inputs sequentially. The GRU layer has a specified number of units, which determines the size of the hidden state.

c) Dense Layer: The output of the GRU layer is passed through a dense (fully connected) layer that maps the RNN output to the vocabulary size. This allows the model to generate the next character in the sequence.

C. Long Short-Term Memory Network (LSTM)

The LSTM is another variant of the RNN that uses a more complex gating mechanism to selectively remember and forget information from the input sequence. This allows the LSTM to handle long-range dependencies in the data better.

a) Embedding Layer: This layer maps the input characters (represented as indices) to dense vector representations using an embedding matrix. The size of the embedding matrix is determined by the vocabulary size and the embedding dimension.

b) LSTM Layer: The recurrent layer is an LSTM that processes the embedded inputs sequentially. The LSTM layer has a specified number of units, which determines the size of the hidden state and cell state.

c) Dense Layer: The output of the LSTM layer is passed through a dense (fully connected) layer that maps the RNN output to the vocabulary size. This allows the model to generate the next character in the sequence.

D. Transformer

The RecipeTitleTransformer module has the following key components:

a) Embedding Layer: This layer maps the input characters (represented as indices) to dense vector representations using an embedding matrix. The size of the embedding matrix is determined by the vocabulary size and the model dimension.

b) Positional Encoding: The positional encoding module adds positional information to the embedded inputs, allowing the model to capture the order of the sequence.

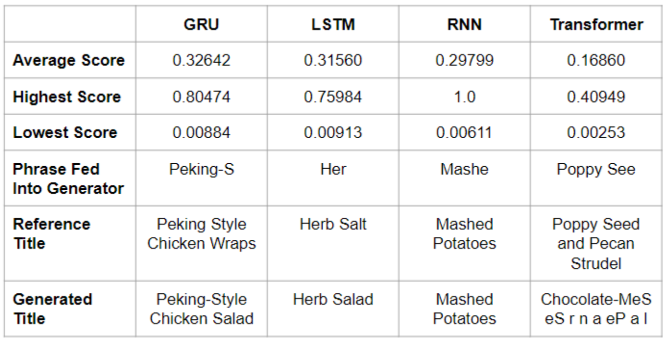
c) Transformer Layer: : The transformer layer is a stack of multi-head attention and feed-forward layers that process the encoded and positionally encoded inputs.

d) Dense Layer: The output of the transformer layer is passed through a dense (fully connected) layer that maps the transformer output to the vocabulary size. This allows the model to generate the next character in the sequence.

IV. Results and Analysis

In order to quantitatively measure the performance of each model, the BLEU score metric was used. BLEU score is calculated by using precision to calculate the number of sequences that exist both in the target and predicted titles. These precision metrics are calculated for sequences of 1, 2, 3, and 4 words. These precision metrics are summated and multiplied by a brevity penalty, which mathematically discourages the model from predicting titles that are too short.

Comparing the BLEU scores of the different models, the GRU model achieved the highest average score of 0.32642, with the highest individual score of 0.80474. The Transformer model had the lowest average and individual BLEU scores. These results can be seen in Figure 2 below:



	GRU	LSTM	RNN	Transformer
Average Score	0.32642	0.31560	0.29799	0.16860
Highest Score	0.80474	0.75984	1.0	0.40949
Lowest Score	0.00884	0.00913	0.00611	0.00253
Phrase Fed Into Generator	Peking-S	Her	Mashe	Poppy See
Reference Title	Peking Style Chicken Wraps	Herb Salt	Mashed Potatoes	Poppy Seed and Pecan Strudel
Generated Title	Peking-Style Chicken Salad	Herb Salad	Mashed Potatoes	Chocolate-MeS eS r n a eP a l

Fig. 2. Table Comparing RNN, GRU, LSTM, and Transformer Results

The distribution of BLEU scores for each model type can be seen in the histograms in Figures 3-6 below:

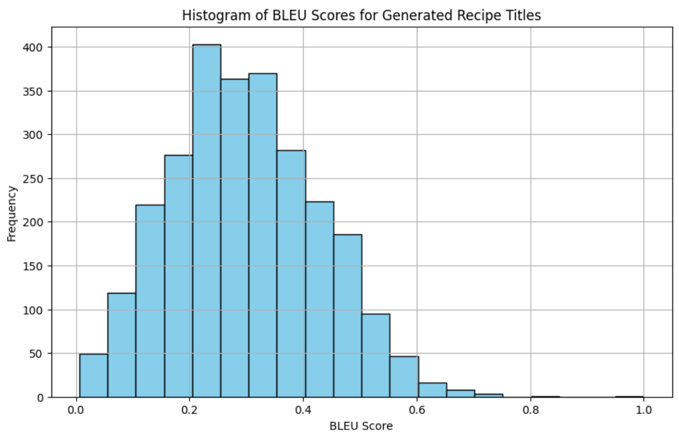


Fig. 3. Histogram of BLEU Scores for RNN Model

The loss curves showed the GRU and LSTM models converged better than the RNN and Transformer models. The RNN loss curves can be seen in Figure 7, LSTM in Figure 8, GRU in Figure 9, and Transformer-based in Figure 10.

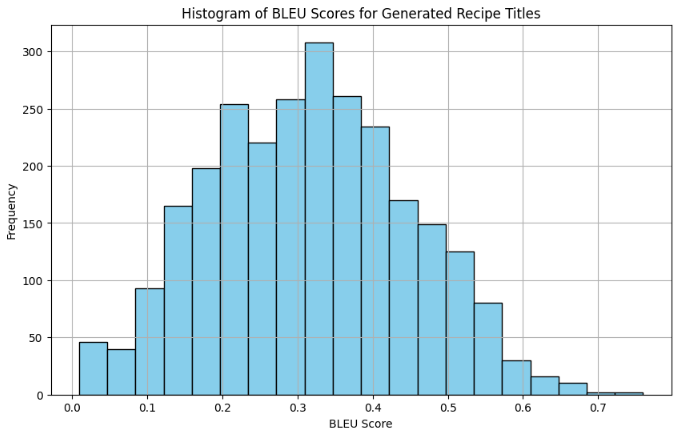


Fig. 4. Histogram of BLEU Scores for LSTM Model

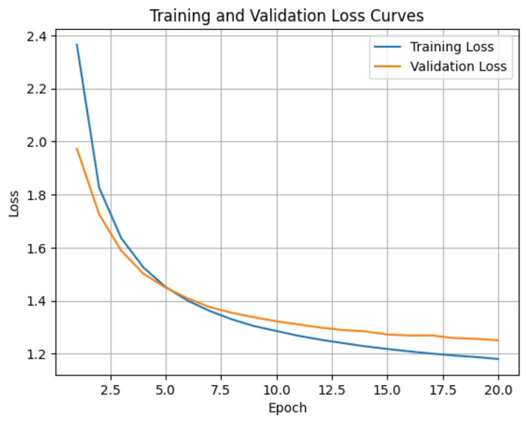


Fig. 7. Training and Validation Curves for RNN Model

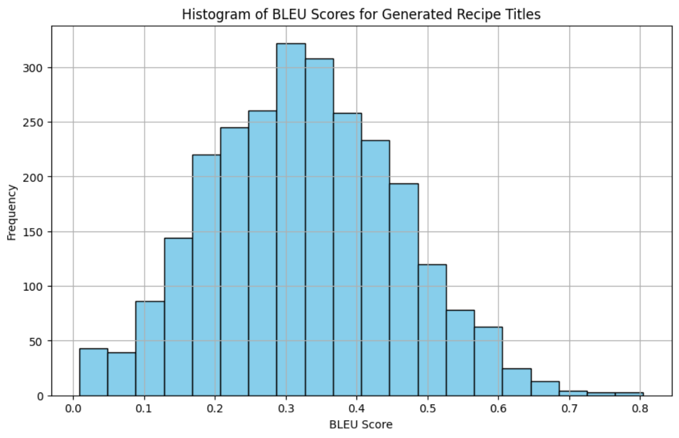


Fig. 5. Histogram of BLEU Scores for GRU Model

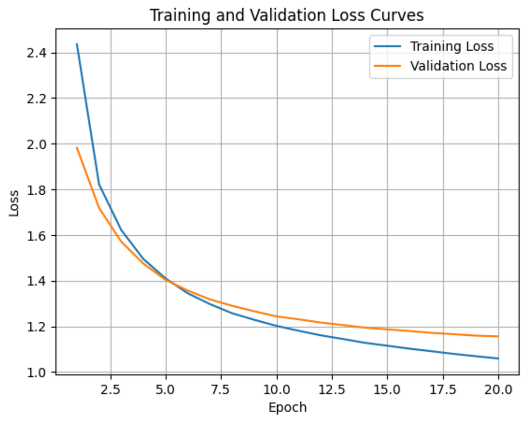


Fig. 8. Training and Validation Curves for LSTM Model

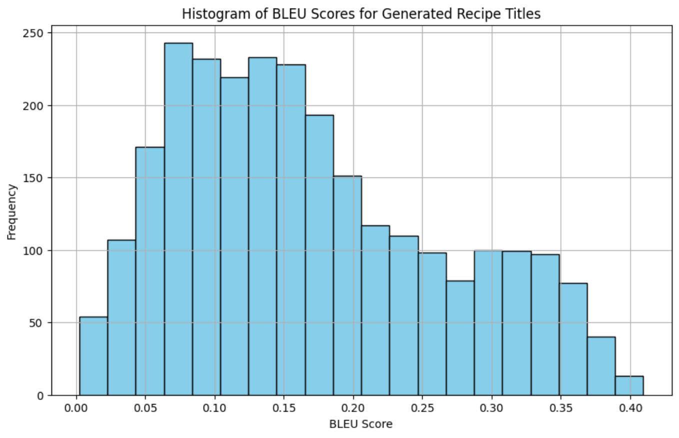


Fig. 6. Histogram of BLEU Scores for Transformer Model

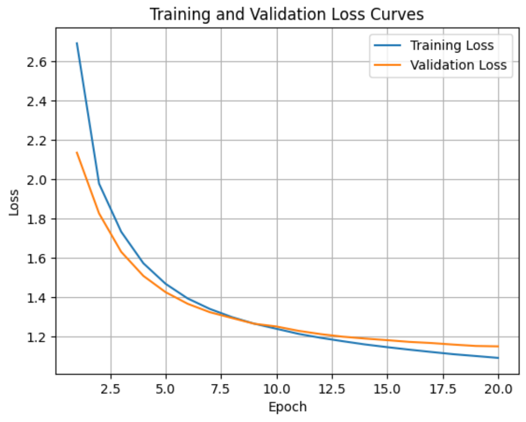


Fig. 9. Training and Validation Curves for GRU Model

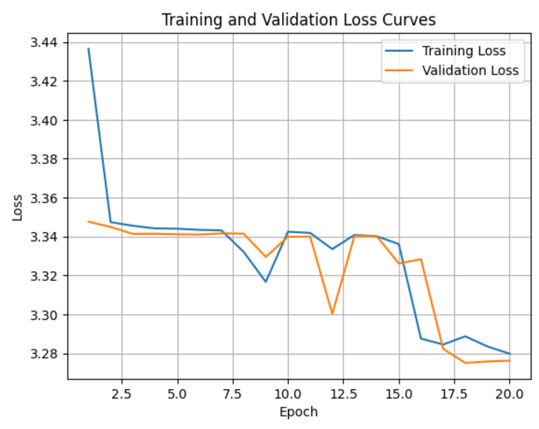


Fig. 10. Training and Validation Curves for Transformer Model

V. Lessons Learned

The key lessons learned include that deep learning is not always the best approach to a problem and that the dataset was highly skewed towards 5-star reviews, lacking sequential information. Potential improvements include expanding to use recipe images and fine-tuning the model architectures. This project demonstrates the potential of deep learning for generating recipe titles. The GRU model achieved the highest average BLEU score, indicating its effectiveness in generating coherent and contextually appropriate titles. However, the dataset's limitations and the model's performance on the test set highlight areas for improvement. Future work will focus on addressing these limitations and exploring new approaches to generate more diverse and accurate recipe titles.

References

[1] Epicurious. (n.d.). Recipe Names. Retrieved from ¡https://www.epicurious.com/recipes/food/views/recipe-names¿ [2] Food.com. (n.d.). Recipe Reviews. Retrieved from ¡https://www.food.com/recipes/recipe-reviews¿ [3] Overleaf. (n.d.). LaTeX Template for Reports. Retrieved from ¡https://www.overleaf.com/latex/templates/report-template¿