Recipe Generation using LSTM

Ashka Shah
University of Toronto
`ashka.shah@mail.utoronto.ca`

Afnan Rahman
University of Toronto
`afnan.rahman@mail.utoronto.ca`

Abstract

This paper proposes an approach to build a model that generates recipes. The model will use the recipes as input sequences and produce new recipes as output sequences. This task requires the model to learn the structure of recipes and the dependencies between ingredients and instructions. In this project, we developed a Encoder-Decoder model with an LSTM and an attention layer. After qualitative and quantitative evaluation we concluded while the model does a decent job generating recipes including the given ingredients, however, further steps such as further exploring methods like the Professor Forcing can be taken improve the model performance.

1 Introduction

In the age of globalisation, people, cultures, and cuisines are more interconnected than ever. With the expansion of the internet, and availability of recipe databases, and cooking apps, individuals now have a wide range of culinary ideas. The easy access to diverse flavours, and ingredients makes it possible for one to experiment with different cuisines in the comfort of their own kitchen. For humans, food is no longer merely a necessity to survive, but a form of art to express their culinary creativity.

The integration of artificial intelligence has resulted in some novel flavour combinations and successful culinary experiments (Frąckiewicz, 2023). It has also enhanced the culinary experience by streamlining meal planning and cooking processes. In scenarios, where one wants to prepare a meal using some leftover ingredients in the refrigerator, they often wonder if by adding only one or two additional ingredients, a favourable dish could be made. But then the question arises what ingredients to add to the grocery list to complete this dish.

In reality, whether it is seeking inspiration for culinary pursuits or trying to make a satisfying meal with some leftover ingredients in the refrigerator, deciding what to cook is a common dilemma. With technology's growing impact on our daily life, deep learning model can serve as a practical solution to our recurrent confusion of what to cook.

The project aims to unravel the mystery behind recipe creation, helping individuals in their culinary pursuits by making the cooking process both efficient and enjoyable.

2 Background and Related Work

In recent years, machine learning, particularly deep learning techniques in the culinary domain, has significantly influenced cooking styles, food trends, and people's engagement with food. These technological advances have revolutionised traditional meal preparation, introducing innovative tools for both professional chefs and home cooks. Tools like food or ingredient detection, or recipe recommendation have received increasing attention in recent years. However, the majority of these deep learning-based tools largely depend on object detection and/or classification (Rokon et al., 2022).

Some recipe recommendation systems have excelled in providing personalized suggestions based on individual preferences, dietary restrictions, and available ingredients. However, they often fall short in ensuring diverse and inclusive representation of various cuisines in their recommendations.

An attention-based convolutional neural network proposed in Jia et al. (2022) is an example of such a recipe recommendations model. It provides tailored suggestions by learning the user-preference of the ingredients, however, the model uses the ‘Chinese Food Composition Table’ to code ingredients. The model architecture is also built to parse mainly recipes from Chinese cuisine (Jia et al., 2022). Therefore, models as such may not perform well for recipes of a different language or cuisines. Lastly, while there are models that can conduct text-based recipe retrieval (Wang et al., 2008), there is not any research in the culinary domain that utilises sequence to sequence (seq2seq) models that can generate recipes given a sequence of ingredients.

To address this gap, the paper proposes the development of a seq2seq Recurrent Neural Network (RNN) model trained on a more comprehensive dataset, containing 125,000 recipes from diverse cuisines from the western world. We understand the dataset is not representative of some well-known eastern cuisines such as Chinese and Indian, however, due to the culinary diversity in the Western counties, the model would be more robustly trained on this heterogeneous data. Our objective is to create a model that generates novel recipes based on user-provided ingredients, serving as a dependable kitchen assistant to simplify everyday cooking.

To capture the long-term dependencies inherent in sequential recipe data, the paper suggests building a Long Short-Term Memory (LSTM) model. This model aims to leverage the inherent patterns and associations within the recipe data to transform user-provided ingredients into robust and diverse recipe generation. While achieving equal inclusivity of various cuisines in recipe data remains a work in progress, the proposed model represents a step towards more well-rounded and globally representative culinary assistance.

3 Data

This section will introduce the data used to train, test and validate the model and the steps taken clean, process and format the data set.

3.1 Choice of Data

For this project the dataset that will be used is the Eight Portions Dataset Lee (2017). The dataset is extremely detailed with each recipe having 4 elements: (1) Title – name of the recipe (2) Ingredients – what is needed for the recipe (3) Picture – image of the recipe (4) Instructions – how to make the recipe. Additionally, measurement units for ingredients are consistent across all recipes.

3.2 Data Cleaning and Formatting

The whole dataset was contained in 3 JSON files. All three JSON data files were merged and converted into a list of dictionaries where each dictionary value contained information about one recipe (title, ingredients, instructions, picture link).

3.2.1 Removing Noise

In order to clean and format, the data was looped through every recipe to remove irrelevant fields i.e. ‘Picture’ and ’Title’. We also removed the recipes that were missing either ‘Ingredients’ or ‘Instructions’. Some of the recipes had strings that didn’t make sense. For example we saw ’ADVERTISEMENT’ and ’SOURCEhttps://’ in the ‘Ingredients’ section, but they shouldn’t be there. We suspect that these strings exist in the data because the recipes were scraped from webpages. It could have been that they were accidentally scraped along with other html elements and weren’t identifiable enough to be removed. All text was converted to lowercase for uniformity and to reduce the inherent bias of the dataset.

3.2.2 Removing Punctuations and Numbers

While in recipe generation, it is unnatural to not have any punctuation or number measurements. However, according to Ek et al. (2020), older RNN models like LSTM are sensitive to changes in punctuations. Thus, we removed any punctions that existed in the dataset. In our first step towards recipe generation, we focused more on the novel combinations of inputted ingredients rather than a perfect output.

We removed the numbers as it is beyond the model's ability to generate its own measurements for ingredients. It can only use the ones that exist in the vocabulary which will not be reliable given the nature of the model. Due to this, the units pertaining to measurements such as 'tablespoons' , 'cups', 'teaspoons', 'ml', 'liter', 'l' etc were also removed.

3.2.3 Data Filtering and Padding

RNN models are commonly known to encounter vanishing gradient problem, therefore we limited the length of the sequence. To do this, we plotted a distribution of the overall recipe length. We determined that 65% of the recipes were shorter than 957 characters. Any recipe greater than that length was discarded from the dataset, leaving us with a dataset of 80898 recipes.

3.3 Vocabulary and Tokenization

We opted for a word level language model as opposed to our suggested character level language model in our proposal. The rationale behind this was that a word based model is better at capturing the sematic relations between the input prompt and the generated output allowing for interpretability. We also wanted to make sure that the ingredients in the input show up in the output i.e. the instructions. With character level models, the vocabulary would be a lot smaller but we would face the issue of generating words that have no meaning.

For the word model, we tokenized the recipes and made a vocabulary list. The original size of the vocabulary that included words from all 80898 recipes was about 90K. With available resources, such a large vocabulary list would be difficult to work with. Therefore, we only included words that appeared atleast 30 times in the 87K recipes. This reduced the vocabulary to 4413 words.

3.4 Generating Inputs and Targets

The tokenized data had two parts the inputs (tokenized ingredients) and target sequences (tokenized instructions). All the words were then converted into numerical indices. Of all the recipes, the maximum input sequence (ingredients) length was 106 and the maximum target sequence length was 148 words. All input and target sequences smaller than these lengths were paded with a `<pad>` token for uniformity. The beginning of the input and target sequences were denoted with `<bos>` and the end of the sequences were denoted with `<eos>`.

3.5 Splitting the Data

The processed data that contained numeric indices of word tokens was split into training, validation, and test set in a 60:20:20 fashion. The sizes each set was: training set: 48538, validation set: 16179 and test set: 16181. The dataset was pickled for direct use in the model. This saved memory and time required to run the data processing.

4 Model Architecture

This section will outline the architecture of our Baseline Encoder Decoder Recurrent Neural Network (RNN) Model and Attention Encoder Decoder Recurrent Neural Network Model.

4.1 The Baseline Decoder Model

The baseline model is sequence-to-sequence LSTM-based Encoder Decoder Model. The model comprises of two main comprises parts (1) an Encoder (2) a Decoder.

Encoder
The encoder has 2 layers in the order:

- **Embedding Layer:** This layer takes in the input sequence and transforms the discrete indices into continuous vectors. Dropout regularization at a probability of **0.1** has been integrated into this layer. The generated embedding of size **256** is then fed into the next layer.
- **LSTM Layer:** This layer has **256** hidden units. The **hidden states** generate by this layer are then fed into the LSTM of the decoder. While the **output states** are discarded.

Decoder
The hidden states of the decoder are initialized from the hidden states of the encoder. The decoder has 3 layers in the order:

- **Embedding Layer:** This layer takes in the target sequence and transforms the discrete indices into continuous vectors. Dropout regularization at a probability of **0.1** has been integrated into this layer. The generated embedding of size **256** is then fed into the next layer.
- **LSTM Layer:** This layer takes the embeddings along with the hidden states from the previous time step. It has 256 hidden units. The **output states** generate by this layer are then fed into next layer, while the **hidden states** are discarded.
- **Linear Output Layer:** This layer produces scores for each token in the vocabulary. Finally, the *Softmax Activation* converts raw scores into log probabilities. The index with the highest probability is the next word predicted.

4.2 The Proposed Attention Model

We want to generate recipes given some input ingredients would require understanding the semantic use of the ingredients (input) the in the recipes (target). An attention-based architecture allows the model to focus on different parts of the input sequence while generating the output, enabling it to capture more complex relationships in the data, especially in tasks where alignment between input and output sequences is crucial. The implemented model has an Encoder that is the same as the Baseline Model. The part of the architecture that differs from the Baseline Model is the Decoder. Here are the details:

Attention Decoder
The Attention Decoder uses an additional Luong Dot Attention Layer. It computes attention weights and generates a context vector based on the decoder's hidden state and the encoder's output. **The hidden states of the decoder are initialized from the hidden states of the encoder**. It has 4 layers int the order:

- **Embedding Layer:** This layer takes in the target sequence and transforms the discrete indices into continuous vectors. Dropout regularization at a probability of **0.1** has been integrated into this layer. The generated embedding of size **256** is then fed into the next layer.
- **Luong Attention Layer:** This layer takes in the decoder hidden states from the previous time sequence along with the encoder outputs. It performs Dot Product Attention scoring and generates a context vector that is fed into the LSTM layer. Luong et al. (2015)
- **LSTM Layer:** This layer takes the context vector along with the hidden states generated from the previous time step. It has 256 hidden units. The **output states** generate by this layer are fed to next layer, while the **hidden states** are discarded.
- **Linear Output Layer** : This layer takes in the output states and produces scores for each token in the vocabulary. Finally, the *Softmax Activation* converts raw scores into log probabilities. The index with the highest probability is the next word predicted.

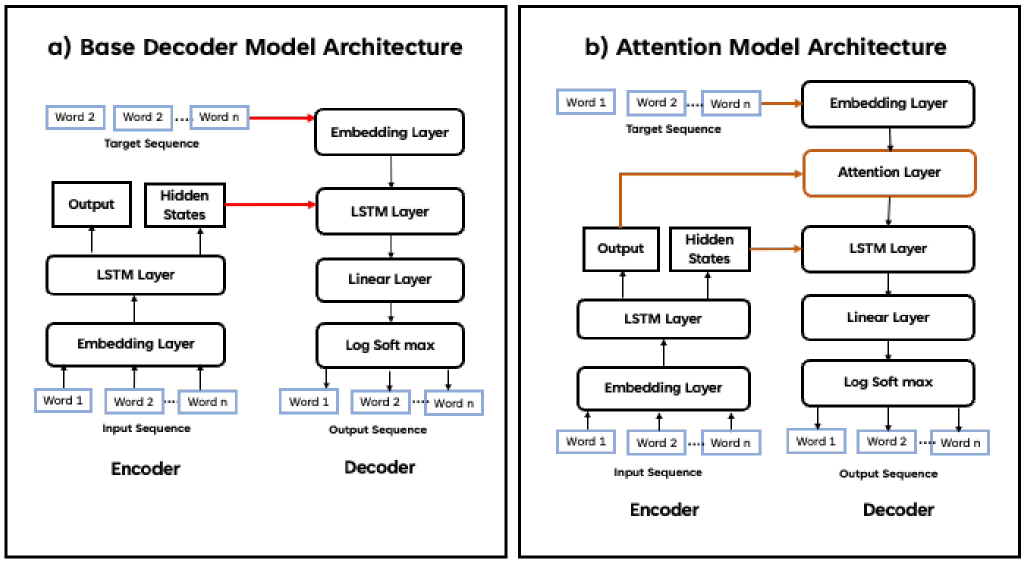


Figure 1: Model Diagrams

4.3 Model Training

This section outlines the key elements used to train our models. Both models were trained using the same mechanisms and parameters.

4.3.1 Hyperparameters

We trained the model using the following hyperparameters values: learing rate (0.1, 0.01, 0.0001), batch size (100,200) epochs (1,2, 5, 7, 10) and dropout(0.1, 0.5). We also tuned the teacher forcing to investigate its effect by using ratios 0, 0.5 and 1. We finally came up to the conclusion that a batch size = 100, learning rate = 0.001, dropout = 0.1 and epochs = 5 is ideal for the model.

4.3.2 Loss Function

The Cross-Entropy Loss function was used along with optimization algorithms such as Batch Gradient Descent and the Adam optimizer.

5 Results

Both the Baseline and Attention Models were trained with three different levels of Teacher Forcing: (1) 0% Teacher Forcing - No teacher forcing used during training. (2) 50% Teacher Forcing -Teacher forcing was used in half of the training instances. (3) 100% Teacher Forcing - Teacher forcing was used on all training instances.

To assess the effectiveness of our models, we conducted both quantitative and qualitative analysis. For quantitative evaluation during the training phase, we used perplexity on the validation set as a metric. Perplexity measures how well the model predicts the next word given some context words Riccio (2023). Thus, higher perplexity values correspond to poorer model performance. Training loss was used along with validation perplexity to identify instances of over-fitting. The calculated training loss, training perplexity and validation loss for each model can be seen in **Figure 2**.

The results obtained indicate that, across all teacher forcing variations, the Attention Model consistently showed a lower validation perplexity when compared to the Baseline Model. Specifically, the Baseline Model trained with 100% teacher forcing demonstrated the highest validation perplexity, implying worst performance. On the other hand, the Attention Model trained with 0% teacher forcing showcased the lowest validation perplexity, quantitatively indicating best performance.

Since this was a generative model, it was necessary to evaluate it qualitatively as well. Human evaluation was used to access the sensibility of the recipes generated by the model. The qualitative evaluation also helped us determine the occurrence of the given ingredients in the generated recipes.

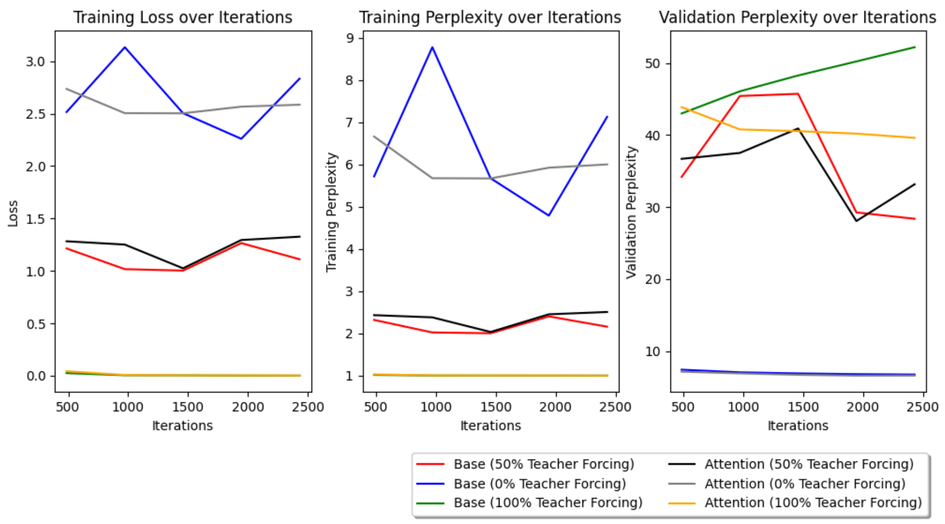
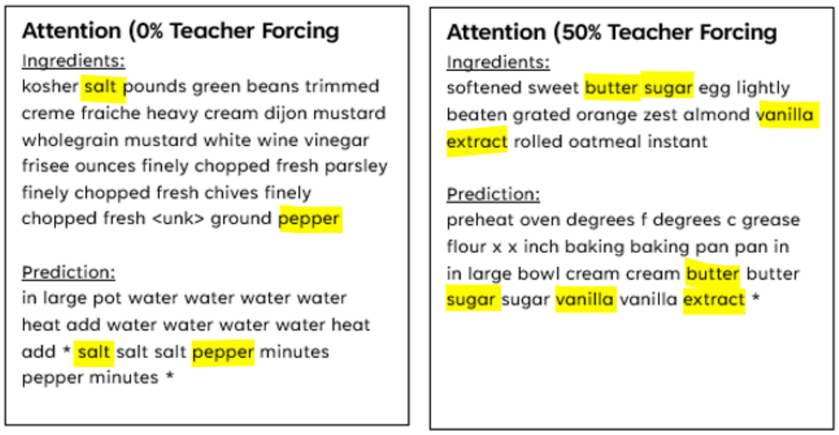


Figure 2: Quantitative evaluation of the Base Model and the Attention Model. Each of the three subplots include three types of results for each model. Every 486 iterations is the end of one epoch.

In **Figure 3 and 4** show a snippet of a recipe generated by each of the four models. These recipes were generated by feeding a randomly chosen test data point to each of the four models. The recipe generated by the Attention Model (trained with 50% teacher forcing) had four given ingredients (butter, sugar, vanilla and extract). In contrast, the base model was unable to generate recipes incorporating as many of the given ingredients. It must be highlighted that both Attention and Baseline Model trained with 100% Teacher Forcing were unable to generate any output when given ingredients from the test set. Additionally, each of the predicted recipes included repetition of words.



Attention (0% Teacher Forcing

Ingredients:
kosher salt pounds green beans trimmed creme fraiche heavy cream dijon mustard wholegrain mustard white wine vinegar frisee ounces finely chopped fresh parsley finely chopped fresh chives finely chopped fresh <unk> ground pepper

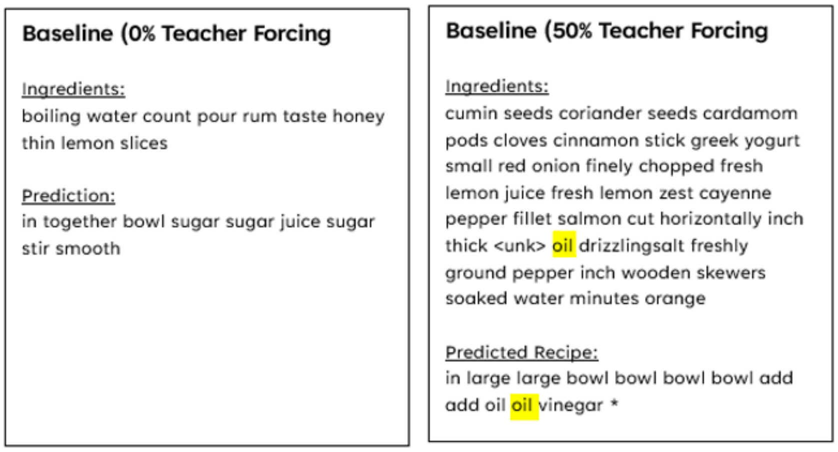
Prediction:
in large pot water water water water heat add water water water water heat add * salt salt salt pepper minutes pepper minutes *

Attention (50% Teacher Forcing

Ingredients:
softened sweet butter sugar egg lightly beaten grated orange zest almond vanilla extract rolled oatmeal instant

Prediction:
preheat oven degrees f degrees c grease flour x x inch baking baking pan pan in in large bowl cream cream butter butter sugar sugar vanilla vanilla extract *

Figure 3: Recipes generated by the Attention models. Note: The * represents repetition of the word. The Base and Attention Model trained using 100% teacher forcing generated no recipes. Thus, they were excluded from the figure.



Baseline (0% Teacher Forcing

Ingredients:
boiling water count pour rum taste honey thin lemon slices

Prediction:
in together bowl sugar sugar juice sugar stir smooth

Baseline (50% Teacher Forcing

Ingredients:
cumin seeds coriander seeds cardamom pods cloves cinnamon stick greek yogurt small red onion finely chopped fresh lemon juice fresh lemon zest cayenne pepper fillet salmon cut horizontally inch thick <unk> oil drizzlingsalt freshly ground pepper inch wooden skewers soaked water minutes orange

Predicted Recipe:
in large large bowl bowl bowl bowl add add oil oil vinegar *

Figure 4: Recipes generated by the Base models. Note: The * represents repetition of the word. The Base and Attention Model trained using 100% teacher forcing generated no recipes. Thus, they were excluded from the figure.

6 Discussion

The results indicate that our proposed Attention Model outperformed the Baseline model in generating recipes. This is due to the attention mechanism, which enables the model to weigh different parts of the input sequence at each decoding step.

In contrast, the Baseline model compresses the whole input sequence into a fixed-size hidden state, potentially causing information loss, especially for the long sequences. In comparison, the Attention Model generated more coherent recipes with the specific ingredients. In our understanding, the attention mechanism's ability to preserve detailed information throughout the sequence potentially have contributed to this Cristina (2023)

The main purpose of analyzing our models' performances with various teacher forcing ratios, was to gain some insights into the two models' behavior. With 100% teacher forcing, we observed high validation perplexity and low training loss which indicated potential overfitting for both models. This hypothesis was later confirmed as both models failed to generate meaningful recipes using test data, suggesting that teacher forcing might hinder proper weight updates thus hampering the learning process. Training with 100% teacher forcing, however, led the Attention Model to incorporate some ingredients which was still an improvement. The base model continued to generate recipes with no incorporation of the specified ingredients.

Training with 50% teacher forcing led to the best results for both models. The attention model had comparatively one of the lowest validation perplexity indicating the model performing well on new data and most likely not overfitting. As a result, among the four models, the Attention Model generated recipe had the highest occurrence of the specified ingredients. The base model's recipe generation also improved in including given ingredients. Given that teacher forcing has shown partial effectiveness in our training, we can explore similar techniques like Professor forcing Lamb et al. (2016) to enhance and further refine the model.

However, these predicted recipes had a short sequence length and included repetitive words. This is a consistent observation across the four model types. We noticed that with increased numbers of epochs, the sequence length increased but the repetitive patterns decreased. That led us to infer that increasing the number of training epochs might mitigate repetition and increase sequence length. However, due to limited GPU power, we were unable to train the model for an extended duration. Future work should train the model on employing larger computational resources.

7 Limitations

The model is unable to generate an end to end recipe. That can occur due to several reasons. The Encoder-Decoder model architecture we used to generate recipes commonly used for tasks such as translation and text summarization. Our model uses ingredients to generate recipes which is quite different than the prompts an Encoder-Decoder model normally performs well on.

The proposed model does not generate semantic texts as we removed the stop words. It is also not able to generate its own measurements. Also the repetition of the words presisted in all attempted versions of the model. As the models were trained using Google Colab, we were also limited by the GPU runtime and were unable to train models for longer epochs.

We had to restrict our vocabulary size during training, that have potentially prevented the model from exploring connections between numerous words. This limitation might have influenced the diversity and complexity of the generated recipes.

8 Ethical Considerations

Ethical issues this model can have is due to the dataset lacking representability of the Eastern cuisines such as Indian, Chinese, Caribbean etc. The trained model may not be as inclusive and representative of other cultures as it is of the Western culture which can be a ethical concerns people from different cultural and religious backgrounds can have different dietary restrictions.

It is also difficult to assess the consumption safety associated with newly generated recipes. Lastly, the model does consider any food allergies the individual has. So, there needs to be a warning/disclaimer for potentially controversial ingredients when the recipe is outputted.

9 Conclusion

In conclusion, our project took a novel approach to utilize an Attention based Encoder Decoder LSTM model to generate recipes. While, this project highlights the strengths of such a model in generating new texts, using short-lengthed prompts like ingredients, limitations of the Model Architecture coupled with our scarce computational resources has opened a new door for future research and model enhancements.

References

Cristina, S. (2023). The attention mechanism from scratch. MachineLearningMastery.com.

Ek, A., Bernardy, J.-P., and Chatzikyriakidis, S. (2020). How does punctuation affect neural models in natural language inference. In *ACL Anthology*.

Frąckiewicz, M. (2023). Ai and the future of food pairing: How algorithms are changing the way we experience flavors.

Jia, N., Chen, J., and Wang, R. (2022). An attention-based convolutional neural network for recipe recommendation. volume 201.

Lamb, A., Goyal, A., Zhang, Y., Zhang, S., Courville, A., and Bengio, Y. (2016). Professor forcing: A new algorithm for training recurrent networks. volume abs/1610.09038.

Lee, R. (2017). Recipe box. Eight Portions.

Luong, T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Riccio, D. (2023). Everything you should know about evaluating large language models. Medium.

Rokon, M. S. J., Kishor Morol, M., Binte Hasan, I., Saif, A. M., and Hussain Khan, R. (2022). Food recipe recommendation based on ingredients detection using deep learning.

Wang, L., Li, Q., Li, N., Dong, G., and Yang, Y. (2008). Substructure similarity measurement in chinese recipes. In *Proceedings of the 17th International Conference on World Wide Web*.