Cooking with BERT
Midway Report

Alexander Schott, William Braga

Abstract

The generation of food recipes is a common machine learning task within natural language processing. It has been undertaken in various ways, most commonly through char and word-level recurrent neural networks. However, upon testing some of the models that have been published online, their shortcomings quickly become clear. Although many of these artificial recipes model the vocabulary and style of cooking instructions, they lack the ability to create plausible directions. Namely, these models lack the nuance to capture ingredient permanence, where a recipe will not introduce or abandon ingredients after they have been listed, and instruction validity, where cooking steps will not be logically impossible (e.g. cook and cool an ingredient simultaneously). To address these limitations, our goal is to create a more effective model and evaluation mechanism. Currently, we have implemented a character and word-level RNN to act as our baseline in testing. In the coming weeks, our plan is to create a BERT model, which should improve recipe generation. We will test BERT against the simpler RNNs using both intrinsic and extrinsic measures to better capture its predictive ability.

1 Goal

Our goal still remains to create a more effective model using BERT. In addition to comparing our BERT model to simpler RNNs using metrics like accuracy, we will be extrinsically evaluating them by manually annotating their recipe generations. We expect BERT to outperform char and word-level RNNs in both accuracy and plausibility.

2 Data Collection

A large collection of data has been collected composed of approximately 500,000 recipes. These recipes are stored in 33 1 MB text documents (Figure 1).

Statistic	**Value**
Number of Documents	33
Average Document Size	1 MB
Number of Recipes	~ 500,000
Number of Unique Characters	109
Number of Unique Words	10623

Figure 1: Data Summary Statistics

For the implemented RNNs, the training data is a contiguous sequence of characters or words, and the label is the next character/word after the sequence. For example, the data includes the lines “Combine eggs and dill relish. Add the rest of the ingredients, except tuna, and mix really well.” Figures 2 and 3 show how the string would be processed by the models.

#	Input	Label
1	“Combine eggs and dill relish. Add the re”	s
2	“d dill relish. Add the rest of the ingre”	n
3	“Add the rest of the ingredients, except ”	t
4	“ the ingredients, except tuna, and mix ”	r

Figure 2: char-rnn processing

#	Input	Label
1	"combine eggs and dill relish"	add
2	"dill relish add the rest"	of
3	"the rest of the ingredients"	except
4	"the ingredients except tuna and"	mix

Figure 3: word-rnn processing

The number of characters/words for each sequence and the step by which the next sequence is shifted are hyperparameters for the models. Figure 2 shows a sequence length of 40 characters and step of 15. Figure 3 shows a sequence length of 5 words and a step of 3.

2.1 Data Formatting

Some data formatting has been done. The recipes were originally used in a recipe recommending system and contain tags related to that system at the start and end of recipes. These strings have been replaced by a shorter "[END]" word to indicate separations in recipes.

In order to prepare the documents for character prediction we made slices of the text of 40 characters long and stored in another list the next character as the target for the model. These characters are then transformed into a one hot encoding corresponding to all of the characters in our alphabet.

A similar treatment was done to the data for the purpose of the word level model. The text is cut into slices, but instead of a one hot encoding, we allow the model to learn word embeddings in its first layer based off of the word to index map we make of our vocabulary.

3 Method

We are developing three models: a character-level RNN, word-level RNN, and a BERT LSTM. The first two RNNs were implemented as a baseline to compare the BERT model in order to highlight intrinsic and extrinsic benefits it may have over the simpler RNNs. We believe BERT will be able to create more accurate and intelligible predictions when compared to prior work on food recipe generation.

To analyze the results, we are conducting intrinsic and extrinsic evaluations. Among all three models, we will be determining their accuracies on the test set. Also, we will annotate the recipes generated to mark their validities. These annotations will show whether BERT's recipes follow a better logical flow than that of the other RNNs.

As of now, we have implemented the char and word RNNs and trained their models. The preliminary training accuracies for them have also been calculated at 55.9% and 75.5% respectively. We plan to tune the hyperparameters more and implement the BERT model before beginning the extrinsic evaluations.

4 Preliminary Model

The preliminary models of the char-rnn and word-rnn, are mostly implemented. The models are able to produce text, but we have not fine-tuned the model or allowed them very much training time yet in order to allow them to produce high-quality text. The success of the models in this project will be in their relative performance to each other after the extrinsic evaluation of text they have produced.

4.1 Initial Results

The baseline models we are comparing against are examples of char-rnn output of recipes found on the internet.

We are using a non-standard metric for evaluation. We are annotating the feasibility of recipes to identify where models fail to grasp important characteristics of recipes in their ability to be followed. Because recipes are short instructions intended to be followed by humans, it should be easy for an annotator to identify instances of a model producing instructions that have impossible steps. A low rate of impossible steps per recipe or words of text will indicate models that better grasp what a human is looking for in a recipe. In our six-week plan, human annotating is one of the last tasks we plan to complete.

4.2 Negative Results

At first, the word-rnn was exceeding the RAM capabilities of Google colab. When the input and label matrices were being generated, the environment would crash before the model could begin training. The matrices' shapes were both dependent on the number of words that have been seen, similar to how the input and label matrices' shapes of the char-rnn are dependent on the number of possible characters. However, the training data contains many more unique words than characters (by a factor of about 100), which is why the char-rnn could run without issue. To resolve this, we implemented a generator in the word-rnn that would split the input data into several batches.

5 Work Division

Alexander setup the colab environment and worked on the data formatting for the recipes. He wrote the code to clean the recipes and add them all to one string. After, he split the data into sequences (as shown in Figures 2 and 3) and created the input and label matrices to train the models. Alexander also setup a model checkpoint to save the progress of training done.

Will instantiated the models and trained them with the input and label data. For the word-rnn, he created the generator function so that the training could be done in batches of the input. On both models, he conducted their initial training, tuning, and checkpoint saving.

We both worked collaboratively on the midway report.

6 Future Steps

In the short term we need to put together the BERT model to verify that we can make it work and have all of the major coding parts in place. After that, a lot of time will need to be spent testing hyperparameters to verify the models are performing to the best of their abilities. Otherwise, the comparisons done between models will not accurately reflect their strengths and weaknesses relative to one another. After that we can collect a corpus of model generated text and annotate it for impossible steps. This task is expected to take a reasonable chunk of time as well, but after that, everything will be in place to conduct the final report.

6.1 Next Model Version

The next model we will build will be backed by the BERT language embeddings. The BERT based model is expected to perform better than our other models both in terms of intrinsic loss and our extrinsic annotation method, as it has managed to pass language generation expectations in many other benchmarks.

6.2 Planned Tests

Our research question will not be answered until we finish our annotation process and can decide which models perform better on extrinsic results. Our expectation is that all models will perform well in terms of reducing loss in the language model, but that BERT will make a significant advancement in terms of creating steps that are typically more feasible than steps produced by the other models as determined by our extrinsic evaluation.

6.3 Timeline

We are still on track following the timeline set by the project proposal. The first two weeks following spring break were dedicated to the extraction of data, the creation of the char and word-level RNNs, and the midway report. Following this, the BERT model, tuning, and annotating still need to be done. The remaining timeline is as follows:

Week 3: Implement and test the BERT model. Finish tuning the hyperparameters for all models.

Week 4: Perform extrinsic evaluations through annotations.

Week 5: Finish all annotations and tests and conduct the final report.

7 Future Task Division

On Week 3, Alexander will be in charge of implementing the BERT model, while Will will be working on the tuning of the hyperparameters. The week after, both of us will be working on annotating the model predictions. Finally, on Week 5, we will collaboratively complete the final report and any other testing that needs to be done.