Health-Aware Recipe Generation using LSTMs

Professor Zachary Pardos
Data 244 Fall 2021

Anirudhan Badrinath
abadrinath@berkeley.edu

Eli Pleaner
epleaner@berkeley.edu

Noah Reiner
noahmreiner@berkeley.edu

1 Introduction

There has been a proliferation of online recipes in recent years. Bon Appetit, New York Times, Serious Eats and countless others have gained serious followings. For serious cooks and amateurs alike, they provide inspiration and guidance for hungry cooks. If you have an ingredient you do not know what to do with, search algorithms will return multiple highly rated recipes that contain it. For those of us who have scrolled through recipes while hungry, the trouble is that the recipes that return often contain ingredients that you may not have access to. One ends up having to go get tamarind, kaffir lime leaves, or some other random, yet necessary ingredient.

Furthermore, there are tremendous limitations in regards to the nutritional values of each recipe. It is quite difficult, when adhering to a restricted set of ingredients, to find a recipe that is sweet but also low-fat. Sometimes, available recipes often do not have detailed nutritional information, which presents an issue in the modern day for those with dietary restrictions.

We propose that the next evolution in this process of generating recipes is to move beyond simply inputting ingredients. To this end, we created a recipe generation approach that accepts a nutritional profile as well as a sample of available ingredients of whatever meal you feel like having. This would not only generate a relevant and sensible recipe but one that was created specifically for your dietary needs. If you have pistachios and yogurt, desiring a low-fat snack, such a recipe generator might create a dipping sauce for you. If you input bananas, but want something high-protein and sweet, it would create a nutty banana cookie or cake.

There are a number of use cases for this sort of model. For someone with specific dietary needs, they could easily adjust a recipe to work for them. For those who are tired of scrolling through pages and pages of recipes wanting to create something new, the model could produce personalized recommendations for exactly what is needed. Some people might use CSA's and end up with random ingredients they don't know how to cook — the model would create a recipe that helps them put those ingredients towards something delicious and nutritious.

Typically, training a text generation model entails using an off-the-shelf method such as a recurrent neural network (RNN) or a long-short term memory model (LSTM). However, given that we wish to additionally condition the recipe on nutritional requirements, which is more than past textual history of the recipe, we augment this standard approach. Using an adapted LSTM model architecture with a fully-connected head, we propose a recurrent model to generate recipes based on user input of ingredients and nutritional specification: calories, protein, sugars, and fats. We trained the proposed model with 5M parameters on a large corpus of over 50,000 recipes for 100 epochs using the industry-standard Adam optimizer.

This model worked quite well, achieving an accuracy of around 32% on the training and validation sets. Although this seems quite low for a standard supervised task, the caveat is that it outputted the correct word out of a vocabulary of more than 20,000 words around a third of the time. Additionally, without exception, it outputted reasonable alternatives to many words in qualitative analysis of produced texts. We perform analyses of the model's internal representation of recipes conditioned on different nutritional requirements.

As a group, we iterated on a number of food and recipe text generated ideas. Ultimately, we decided that there is tremendous opportunity in the food space to create recipes that are able to accommodate limited ingredients and a number of dietary needs. We challenged ourselves to create something novel that had hardly been explored. In the end, we created something with financial potential that could create a usable product to make people's gastronomical lives easier and more interesting.

2 Dataset

The primary dataset we used was collected in a collaborative effort between the Universitat Politecnica de Catalunya and the Massachusetts Institute of Technology. The dataset is a collection of over 50,000 recipes, including 325,702 sets of instructions. In total the dataset has a 22,649 word vocabulary. This dataset has been used in quite a few marquee papers on recipe generation using images and vice versa using generative adversarial approaches and autoencoders.

Although the dataset was large, there were only a few features, which were largely clean and organized. That included the recipe title, recipe instructions, ingredients, nutrition per ingredient, nutrition per 100 grams, and the URL. We discussed the interpretability of these features, specifically the per 100 gram features. Although the format was quite usable, people don't generally know or use weight per ingredient or nutrition value per 100 grams. This was one limitation of our dataset which we tackled using a clustering approach to generate nutritional profiles.

The given format was not usable for input for model training, so we had to clean the recipe instructions, which were located in a list within each row. After this, we tokenized every vocabulary word in our trained dataset using NLTK in order to have it be in the proper format to use as model inputs. Since a one-hot representation would entail a vector of length around 20,000 for each word in a recipe, we used word embeddings, which simply output the positionality of the word in the vocabulary (i.e. 42 if the word *banana* was 42nd in the vocabulary).

We show basic statistics and visualizations as part of exploratory data analysis to orient the reader with the dataset and to set reasonable expectations for the model outputs given the quality and quantity within the dataset. Upon inspection, each instruction in a recipe is, on average, 65 characters and 4.57 words long, with 4-8 instructions per recipe. Putting those together, each recipe used in training is expected to be around 15-40 words long; there isn't a great level of culinary detail involved. As a result, we cannot expect the model outputs to be extremely detailed since the inputs are likely quite vague themselves (i.e. combine all the ingredients).

From Figure 1, it is clear that the instructions themselves have a quite large leaning towards baking in a similar manner. Some interesting inclusions are lemon juice, peanut butter, and ice cream, which are intermediate ingredients in a quite a few recipes. As a whole, this figure illustrates that a model that is fitted on this dataset is likely to be prone to picking recipes that involve baking more often than not.

From Figure 2, it is clear that there is a roughly uniform spread of calories throughout the recipes, but there is significant skew in the distribution of fats, sugars, and proteins. That being said, the tail for the sugar and fat distribution is far longer since it's more likely that desserts fit into the category of having high fat, high sugar, and high calories. The same does not hold for a high-protein meal. From this, it's reasonable to

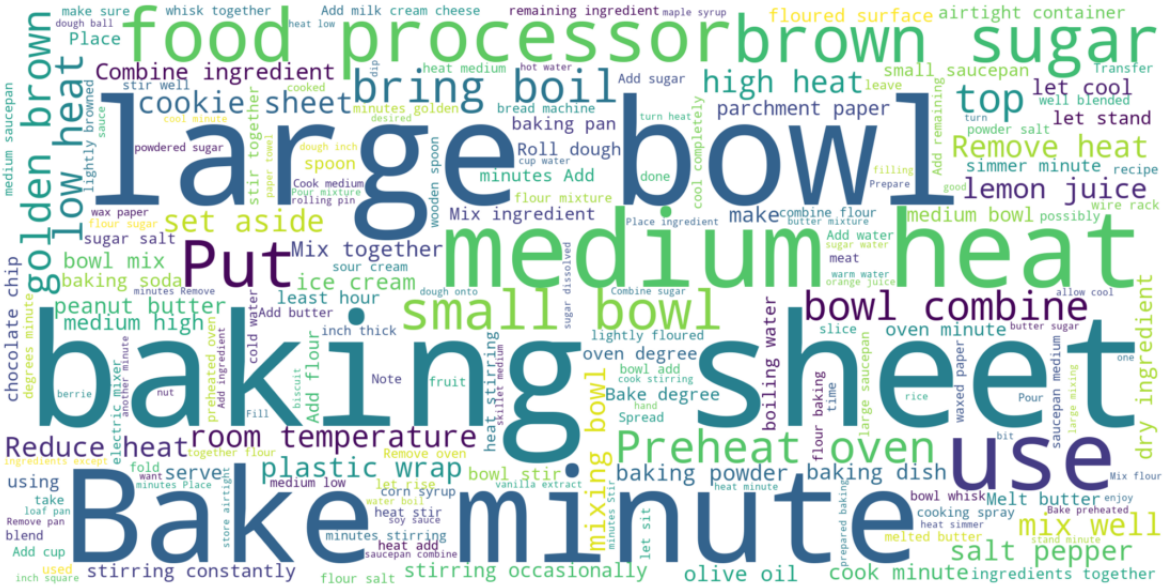

Figure 1: Word cloud of most common words occurring in the recipe instructions in aforementioned MIT CSAIL recipe dataset.

assume that a model will be able to produce recipes with variance in calories, but it might perhaps struggle with high-protein requirements due to clear class imbalance.

The second dataset that we used to support our generation pipeline was found through Kaggle. This dataset is a massive list of recipes, including the rating, ingredients, and title for all of the 11,380 recipes. We mainly utilized this dataset as a source of common recipe instructions to be fed as inputs to our generative model. By using only the first sentence (or a portion of the first sentence) of each recipe, we had a huge list of real-world food-related inputs to be used to generate novel recipes.

3 Methods

We wanted to generate recipes, which is a form of a structured text with logical ordering. We have studied techniques that can generate such sequences of words or characters in the form of recurrent neural networks or long short term memory modules. These have worked quite well in text generation when trained on unsupervised collections of natural language datasets, whether it's Shakespeare like Romeo and Juliet or Wikipedia.

We propose a novel extension to those traditional techniques. We wish to condition our recipe's next word on not just the previous history of words but also the nutritional requirements of the user. Given a neural network framework parameterized by θ, we wish to augment the probabilistic generation of the next word based on all past words, $f_\theta(\mathbf{w}_{t+1} \mid \mathbf{w}_{1..t})$, to incorporate nutritional information: $f_\theta(\mathbf{w}_{t+1} \mid \mathbf{w}_{1..t}, \mathbf{n})$.

In this case, we choose to designate two distinct sections of the model: $f_\theta(\mathbf{z} \mid \mathbf{w}_{1..t})$, which generates recurrent features given past words, and $h_\theta(\mathbf{w}_{t+1} \mid \mathbf{z}, \mathbf{n})$, which combines these features with nutritional requirements to generate the next word in the recipe. As a whole, we propose training f_θ and h_θ end-to-end by leveraging the generalization capabilities of neural networks and LSTMs in Keras.

In terms of neural network architecture, we keep the number of layers quite low to limit the computational complexity. With the two inputs being the nutritional requirements, $\mathbf{n}$, and the past history of the recipe,

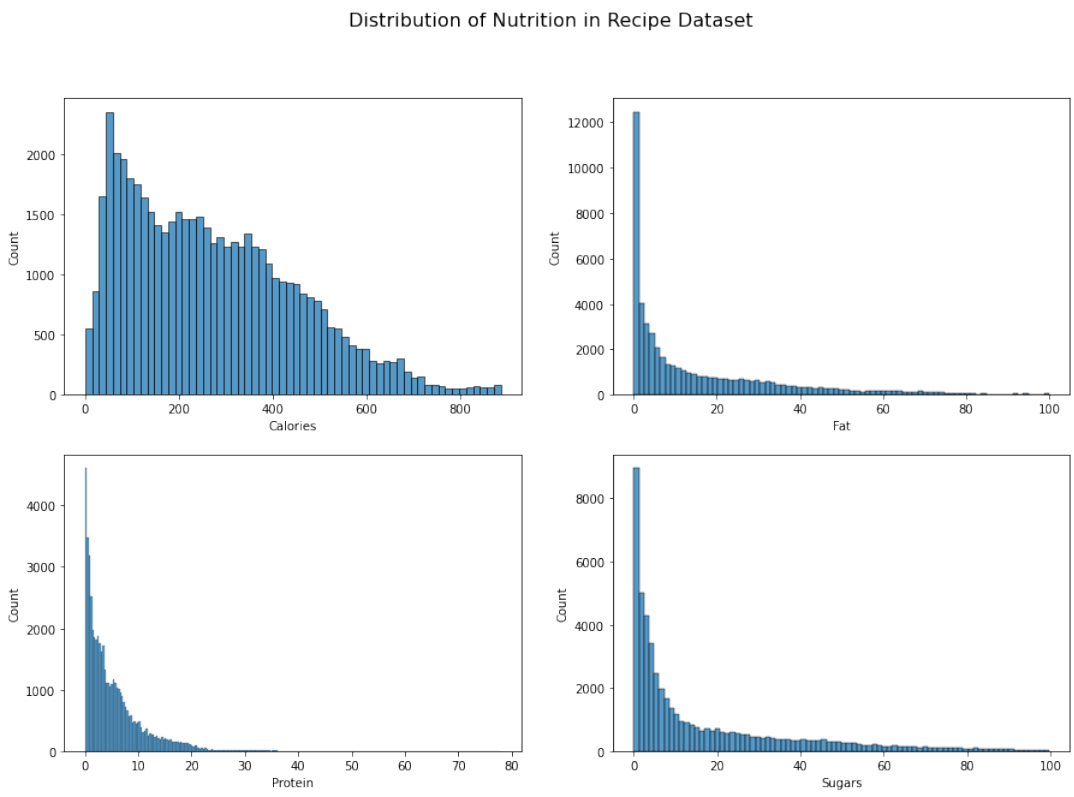

Figure 2: Distributions of calories, fats, proteins, and sugars per 100g in aforementioned MIT CSAIL recipe dataset in the order: top left, top right, bottom left, bottom right.

$\mathbf{w}_{1..t}$, we branch into the two sections described above. To generate features based on the past history of words, we leverage an LSTM layer with a hidden size of 128, referring to f_θ: we refer to this output as "recipe features". We pass these recipe features and the nutritional requirements into h_θ, which is a fully-connected layer of size 22,049, which is the size of the vocabulary. As such, our output is a probability distribution over $\mathbf{w}_{t+1}$; to accomplish this, we apply the softmax function.

Concerning the hyperparameter configuration, we kept it fairly simple because the simple model produced the best outputs. We used a categorical cross entropy loss, which is standard for classification tasks. Industry standard gradient descent techniques like Adam were used and we made no modifications to learning rate or any "history" parameters in Adam since the defaults worked quite well. In summary, we end up with 6,042,973 trainable parameters, with around half coming from embedding the input tokenizations and the other half coming from h_θ. f_θ contains only 100,000 parameters. We show the completed recipe modeling framework as a model architecture figure (Figure 3) below.

Another additional modeling tool that we discovered would be a practically useful element in this quest for practical and healthy recipe generation using artificial intelligence was a categorization of typical nutritional profiles. We hypothesized that there would be around 5-6 general profiles of foods that one consumes throughout the day, and within these individual profiles, there would be a roughly similar distribution of nutrition. As such, one would expect around 5-6 distinct clusters to appear from a K-Means clustering of the nutritional information.

We develop an additional clustering model using standard K-Means that takes the same nutritional information $\mathbf{n}$, and we analyze 6 interpretable clusters of what we consider as discrete nutritional profiles in standard American cuisine. We make no adjustments to existing K-Means hyperparameter configuration

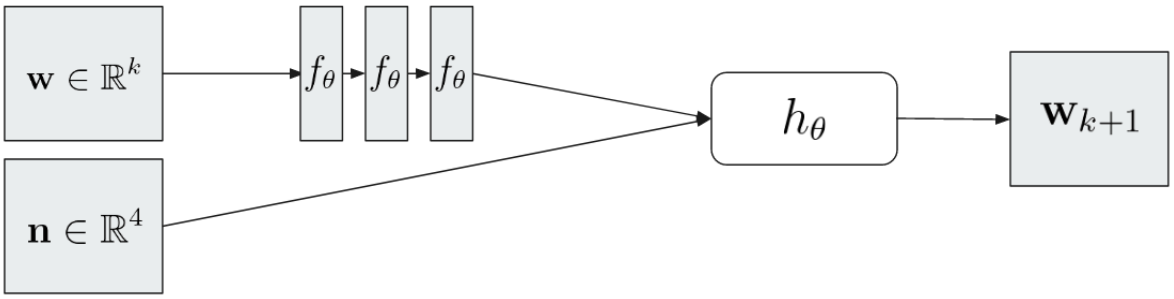

Figure 3: Recipe generation modeling architecture with f_θ describing the recurrent LSTM that generates recipe features and h_θ describing the fully-connected layer that outputs a distribution over the next word.

such as the number of iterations since we discovered that it was near optimal for this task compared to other methods. For reference, the more parameterized probabilistic approach of the Mixture of Gaussians (MoG) performed within margin of error in terms of sum of squared errors (or distortion); hence, we use the simpler non-probabilistic approach of K-Means.

4 Individual Contributions

4.1 Anirudhan's Contributions

I was one of the primary contributors to the data cleaning and modeling, aside from more minor contributions to this report and the final presentation.

In terms of data cleaning, I downloaded the dataset from the CSAIL website, processed and tokenized the dataset to prepare it for training. I produced preliminary visualizations as part of exploratory data analysis, including a word cloud and 4 univariate distributions that would indicate the level at which our model would generate recipes. After deciding that the recipe dataset was simple enough, yet with enough data points to train a neural method, I designed a neural network model architecture with a recurrent framework that incorporated the additional conditioning of the nutritional requirements that were provided in the dataset.

Further, due to the size of the dataset, I implemented a data generator using Python lazy generating functions. Given that we needed to subsample consecutive chunks of the test for training (i.e. the 2nd and 3rd word to generate the 4th, 3rd and 4th to generate the 5th and so on), we were limited to a small batch size of 5 recipes. All in all, this process took quite a while to fine tune, including appropriate pre-padding, tokenizing and so on.

I coded up the resulting neural network framework that conditions the LSTM on nutritional requirements using Keras. Initially, there were thoughts of using SKLearn, but it can neither use GPU nor have multiple input heads or LSTM cells. As a result, we used Keras, which is an easy-to-use deep learning Python library that has been used in class before. This process involved quite a bit of fine tuning with regards to the embedding layer since we could not feasibly include a 20,000+ length vector with simply 1 non-zero value for every word in a recipe.

After successfully training, I developed a simple greedy search function that selected the next word with the highest probability, took that as the ground truth, and repeated this process. This followed any standard autoregressive modeling scheme. There were discussions of developing beam search, but it ended up being too difficult to implement within the given time frame. This greedy search performed quite well. As an extension, I was planning on implementing an ϵ-greedy beam search which would select a random word in

the top 5 next words to provide some increased entropy to the results. This could potentially get the model out of the "rut" of predicting the same sequence of words consistently, which happened quite often since the recipes were short.

I performed some analyses of the intermediate model representations, specifically about the trajectory of the internal representations for both low-calorie and high-calorie recipes (i.e. hidden representations of the neural network). While the high calorie seems a little bit out of distribution for that particular ingredient and nutritional requirement, it generated an interesting visualization animation that showed two distinct internal representation trajectories for both low-calorie and high-calorie variants of the same input text. I used t-SNE and saved multiple pictures of plots to create a GIF visualization showing the evolution of these trajectories.

Following Eli's work on the clustering, I tried to verify using the Mixture of Gaussians, which is a superior probabilistic technique with more parameterization, that the clustering was good enough. As it turns out, we got quite similar results with both a simple deterministic K-Means clustering as the MoG clustering, so we stuck with the K-Means clustering since it was simpler.

Finally, as part of the modeling, I put the nutritional profile and modeling section together by completing a run-through of the clustering by selecting a nutritional profile, dessert for example. Then, I selected an ingredient, vanilla, and let the model generate a recipe. It generated a standard vanilla cake/cookie recipe (somewhat unclear because the ingredient list isn't completely available due to dataset limitations). In a sense, I demonstrated the real-life machine learning pipeline one would use to get a personalized recipe: punch in a desired "cluster" or nutritional profile with available ingredients and get the desired outputted recipe. This worked surprisingly well, but it broke down with a few other categories (dinner, lunch) since those clusters likely had too much variance with too few samples. Given the class imbalance of baking recipes as shown in Figure 1, it was largely inevitable that this could happen.

After this, I had a few smaller contributions to the modeling section of the final presentation and this report. This included generating the figures presented for data cleaning, EDA model architecture, and so on (Figures 1-3), writing the analysis of model's intermediate representations, and minor contributions elsewhere.

4.2 Noah's Contributions

After watching GSI Kevin Moy's presentation and how he used the Python Beautiful Soup library, I knew I wanted to incorporate it in some way. When we began working with our initial dataset, I thought it would be interesting to find clusters of ethnic cuisines within our generated recipes, but the issue was that I couldn't actually find a single table or dataset with enough cuisines and ingredients. So I began using (first learning how to actually implement) the Web scraping library. I scraped a few websites and created a data set of different ethnic cuisines and their ingredients. I gathered and cleaned a few different ethnic cuisines (Indian, Italian, Chinese), but it did not make sense given our goal. It was an annoyingly time consuming process and ultimately was not useful for a text generating model.

Using the secondary data set, I tried a small experiment using the ethnic cuisines I'd web scraped: I ran k-means clustering to see how the clustering might work on our recipes and if it would properly cluster on the ethnic cuisines. It generally worked pretty well, but ultimately was something that was not worth pursuing because of the tediousness of web scraping and then cleaning those results.

Most of the time I spent on this project was in trying to implement a beam search algorithm to improve the results of our model. The idea was that our model was using a "greedy search" algorithm, and after the given input, would pull the next following word with the highest probability. This worked pretty well, but at a certain point, the word with the highest probability would be one the model had already outputted, and thus would end up with a bunch of repeated words (Put the buttercream, put the buttercream, put the buttercream...). With a beam search method, the model would follow a number of different possible paths,

not only considering the highest probability, but the 3 (for example) highest probabilities, pruning the lower chance outputs. Eventually, the algorithm would return the sequence with the highest log likelihood.

I was able to get the log likelihood for the highest probability sequence, but when adding additional sequences, storing the indices and probabilities of every new word became a quite confusing task. Because the model required constant encoding and decoding, it was incredibly difficult for me to keep track of all of the indices and corresponding probabilities. Eventually, I was able to figure out track the probability and indices of the current sequence.

Unfortunately, I had a lot of trouble getting this method to actually work. I believe I was close, but I was not able to produce outputs from our model that would work as inputs for the beam search. It was particularly frustrating spending so much time on something that ultimately did not actually contribute to our final product.

After the beam search experiment, I spent the remainder of our project time trying to generate decent model outputs, adjusting the inputs and nutritional values. Some of what I generated were very useful and interpretable results, and some were complete gibberish or repeated phrases. Furthermore, Anirudhan, Eli and I wrote a script describing our project, from ideation to model processes to analysis of our outputs so that our presentation would be smoother and so that any one of us could present any particular part of the presentation. Lastly, I wrote and organized large parts of this report.

4.3 Eli's Contributions

My responsibilities for this research project include the model training process, nutrition profile clustering and visualization, as well as streamlining an algorithm for generating large swathes of novel recipes. I also contributed to the writing of this research paper, our project presentation, as well as the creation of a consumer-appealing narrative around our research efforts.

After Anirudhan developed the algorithm for training the LSTM model using batch subsampling, I used my Ubuntu machine with a Gigabyte GTX 1080 GPU to train the model on 100 epochs. In order to do this, I used a Docker container set up with Jupyter and a GPU-enabled Tensorflow and Keras branch. Ensuring that Keras was using the GPU for training required some time and configuration.

Once set up, I found that the training process would often run into resource limitations, namely running out of GPU memory. In order to streamline the training process and avoid tedious re-running of the entire notebook after an out-of-memory termination, I wrote the model to disk in HDF5 format every ten epochs. This allowed us to pick training up without losing much progress.

After successfully training for 100 epochs and achieving a satisfactory accuracy rate, I then spent some time exploring input texts that would produce useful generated recipes. I was looking for inputs whose outputs made sense and illustrated our project aims well. However, this proved to be difficult, as it quickly became a guessing game to come up with accurate nutrition information as input. Realizing that no one knows their ideal nutrition content per 100 grams, I set about producing useful nutrition profiles to be used as accurate inputs for recipe generation.

In order to produce these nutrition profiles, I trained a K-Means model on the nutrition columns of our dataset. Clustering this 4-dimensional data (calories, fat, protein, and sugars) into five clusters and looking at the cluster centers, I labeled each cluster for its corresponding "meal". For example, the cluster with high sugars I labeled as "Dessert", while the cluster with lowest calories I labeled as "Snack". To ensure the accuracy of these clusters, I looked at the members of each cluster whose pairwise distance from the center was minimum, as these were the members that best represented the cluster.

In order to visualize these clusters for our presentation, I first attempted to use PCA to reduce the dimen-

sionality from four to two features, and then clustering this reduced-dimensionality model. Visualizing this as a simple scatter plot showed that the reduced-dimensionality clustering did not accurately convey the higher-dimensionality clusters. I then approached dimensionality reduction and visualization using t-SNE, which produced a much more coherent clustering visualization. Tweaking the graphical configuration of the scatter plot, I was satisfied with this visualization, and felt it accurately demonstrated the clustering of nutritional profiles.

Now that we had these nutritional profiles to use, I set about creating a function to automatically generate novel recipes using existing inputs. To do this, I found another recipe dataset from Kaggle, and scraped the first sentence of every recipe to be used as potential inputs. Taking subsamples of these recipe snippets, I iterated through each nutrition profile to generate five recipes for many randomly sampled inputs. This provided us with a streamlined method to gather useful generated recipes, and to compare the effects of nutrition on the generated recipe content.

Finally, along with contributing to this paper and to the presentation, I focused on defining potential use-cases for this algorithm. I sought to answer the question, "why is a nutrition-informed recipe generator useful?" and created user scenarios to illustrate why this generator would be important.

5 Results and Conclusions

We are quite proud of our final results. Despite the limitations of LSTMs as not being the pinnacle of sequence-to-sequence modeling and the computational complexity restrictions, we were able to generate interpretable texts that were interesting variations of recipes we had seen. Our model created sweetbreads with carrots, cookies with strawberries and bananas, and hundreds of other quirky creations (and some inedible monstrosities as well). Overall, it generates recipes that are reasonable and practical to imagine cooking, and while some are likely not as edible as others, they are not bad in general.

We achieved an accuracy of around 32% after 100 epochs training on a GPU using Keras, and while that might seem a bit low, it's quite high in the context of the prediction task. Given that the vocabulary size is around 25,000, it's selecting exactly the correct word more than 30% of the time. And given that, it's likely choosing an appropriate but not exactly correct word for quite a few other instances as you'll see in a few slides. That's actually exactly what we want - it demonstrates understanding of the semantics in recipes. We demonstrate that the nutritional requirements have a significant impact on generated recipes, and we further examine the model's internal representations to look into this.

Further, we generate coherent nutritional profiles that divide into the following qualitatively chosen categories per their cluster centers: breakfast, snack, lunch, dinner, and dessert. We describe and display the utility of generating such discrete nutritional profiles, and we break down and visualize the existing dataset per this clustering.

5.1 Analysis of Model Outputs

We demonstrate the usage of our model to generate coherent recipes conditioned on nutritional requirements. To evaluate the recipes, we qualitatively judge the quality of the recipe in terms of its practicality, cohesiveness, and likely flavour. For judging the flavour, we will be slightly generous given that most of the recipes we will generate will be out-of-distribution in the pairing of nutrition and ingredients. As such, we are bound to end up with some sections of the output being nonsensical. Throughout this generation, we redact repetitions in the model as denoted by three ellipses in the text.

As a preliminary example of a generated recipe, perhaps we have carrots available and we want to make a recipe that has a similar nutritional profile to a low cholesterol cookie. We generate the recipe word-by-word

based on the word with the highest probability conditioned on the past history and nutritional requirements. Note that this is a purely greedy search of recipes, and we do not perform a beam search that could be even more effective.

> "Chop the carrots and artichoke hearts into the triangle of the supermarket. The[y] will be very dark brown in the oven. Preheat oven to 425F (160C) [...]. The bread crumbs, combine the flour, sugar, baking soda and salt in a large bowl and mix well. Add the oil and mix well. [...] Add the rest of the ingredients and mix well with a mixer or a silicone mat or [...] silicone baking mat [...] or nonstick oil."

The produced text is clearly sensible, and it is possible to follow this recipe to end up with a meal. That being said, this is clearly leading towards a baked item (i.e. a cake or cookies) with carrots and artichoke hearts. However, there is clear indication that the model has conditioned on the style of ingredients available (i.e. in using artichoke hearts along with the provided carrots) and that the model has adapted the nutritional values of low-cholestorol sugar cookies by providing a baking recipe.

We extend this analysis to examine the quality of our model's conditioning on nutritional requirements. As a control, we generate a default recipe for banana and strawberry with a healthy nutritional profile (300 calories, 10 grams of fat, 25 grams protein, 5 grams sugar). The model generated a standard and reasonably tasty smoothie recipe with chocolate chips:

> "Put banana and strawberry into the blender or food processor and process until smooth. Add the chocolate chips and stir until smooth [...]. Refrigerate for at least one hour before serving."

With the same input and a reduced number of calories, the model replaced the chocolate chips with the following instructions: `Serve with whipped cream [...] or yogurt.`. Further, with an increased number of calories, the output is as follows:

> "Put banana and strawberry into the blender or food processor and process until smooth. Add the oil and stir to combine. Add the rest of the ingredients and mix well. Shape into balls and place on a greased cookie sheet and bake for 30 minutes or until golden brown. Remove from oven and let cool on a wire rack for 30 minutes before serving. Granola will keep for up to 6 months."

It has converted what was a smoothie recipe into some kind of baking recipe, which coincides with the nutritional requirement to increase the number of calories. Further, the recipe sounds practically doable and sounds reasonable to eat. As such, we demonstrate the model's versatility in generating cohesive recipes with proper conditioning on nutritional requirements.

5.2 Analysis of Model's Intermediate Representations

An extended analysis of the neural network model's internal representation can reveal some insights into its so-called thought process when generating the sequence of words to form a recipe. Specifically, we wish to examine the internal representation as a function of varying nutrition requirements, which is one of the novelties of our modeling framework.

We apply a similar experiment to the previous section in which we generated recipes conditioned on low and high calorie counts. In this case, we generate a control recipe based on carrots with "default" nutrients

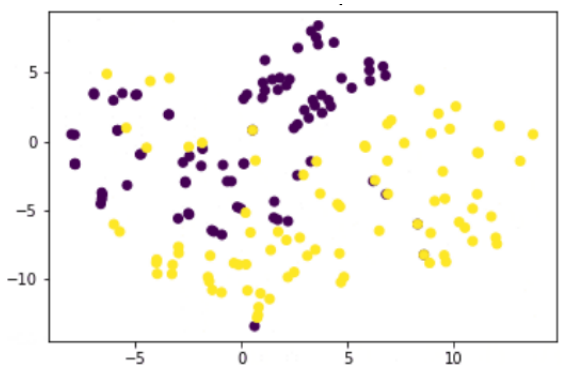

Figure 4: Internal representations of word-by-word generation of high-calorie recipe (denoted by purple dots) and normal-calorie recipe (denoted by yellow dots)

(i.e. average for the training set), and we obtain some yoghurt-oat-carrot-artichoke heart baking procedure as shown below:

> "Chop the carrots and artichoke hearts in a large bowl. Combine the oats generously, seeds [...] in a large bowl, and stir together the wet ingredients and mix well. Add the yogurt mixture to the dry ingredients and stir to combine. Spread the mixture into an even layer on a baking sheet and bake for 30 minutes or until golden brown. Remove from oven and let cool completely before cutting into bars and store in an airtight container for up to 6 months."

When we increase the calorie count, we wish to visualize the differences in the internal representations. However, given that the representation is of length 128, we use t-SNE to reduce its dimensionality down to 2. We display this in Figure 4, where it is clear that the high-calorie recipe generation word-by-word and that of the normal-calorie recipe forms a distinct internal representation trajectory. The higher-calorie recipe is consistently above in the upper left region, whereas the opposite is true of the normal-calorie recipe. As such, we demonstrate that there is valuable internal conditioning on nutrients as we would have hoped for.

5.3 Analysis of Clustered Nutritional Profiles

Through a K-Means clustering with k=5 clusters, we determined five main recipe profiles, as shown in Figure 4. Looking at these cluster centroids, there were clear real-world correlations to the feature values of each centroid. Cluster 0 represented recipes with moderate calories and high sugar content, thus we called this cluster "Dessert". Cluster 1, with low calories and relatively low nutrient content was "Snack". Cluster 2, with high calories and fat, was "Dinner". Cluster 3, with low calories and high sugar, was "Breakfast". Finally, cluster 4, with medium caloric content and high fat and sugar, we labeled as "Lunch". These clusters are visualized in Figure 5, using t-SNE to reduce the dimensions from four to two.

Granted, these clusters were not perfect. Given such a large number of recipes and such varied nutrient content, it is unlikely that five clusters accurately distinguished nutrition profiles in an extremely accurate way. However, for our use case, gathering reasonable nutrition values for recipe generation, these clusters worked just fine.

	Calories	Fat	Protein	Sugar
0	343.573550	15.704199	6.356641	30.464799
1	78.182223	2.224776	3.335685	6.815104
2	681.896008	71.079306	3.320167	6.186558
3	209.804533	7.330212	6.097586	15.589815
4	487.429181	36.610719	6.662075	22.422044

Figure 5: Features of k=5 cluster centroids.

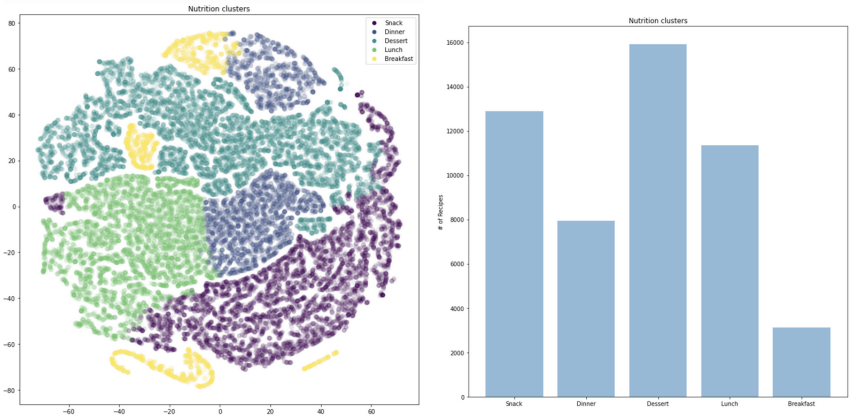

Figure 6: Visualization of nutrition clusters using t-SNE, with distribution of each cluster across dataset.

5.4 Real-World Implications

Imagine being able to look at a touch screen on your refrigerator, which shows the available ingredients in stock. All you have to do is select the nutritional cluster you want: snack, dinner, dessert, etc.; and the model we've created produces a new recipe based on your available ingredients. In a perfect world, that's how we would utilize the model we created. It has largely delivered on those promises despite training on an imperfect dataset with an imperfect model architecture.

When we first began ruminating on this topic, we talked about some of the limitations of the food blogs we liked, largely related to the nutritional limitations we've covered above. In applying our model to these food blogs and recipe sites, the New York Times Cooking, Bon Appetit, etc. would be able to not only focus on quality recipes, but creating unique content for all of their users.

6 References

Kaggle, Recipe Ingredients Dataset, 2016, Version 1, https://www.kaggle.com/kaggle/recipe-ingredients-dataset. Accessed November 2021. Dataset

Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. Marin, Javier and Biswas, Aritro and Ofli, Ferda and Hynes, Nicholas and Salvador, Amaia and Aytar, Yusuf and Weber, Ingmar and Torralba, Antonio, 2020, http://im2recipe.csail.mit.edu/. Accessed November 2021. Dataset.

Trekhleb, Oleksii. "Generating cooking recipes using TensorFlow and LSTM Recurrent Neural Network: A step-by-step guide.", Towards Data Science, 18 June 2020, https://towardsdatascience.com/generating-cooking-recipes-using-tensorflow-and-lstm-recurrent-neural-network-a7bf242acad3, Accessed November 2021.