Project Final Report Form

20 points, Limit 10 pages

Project title	bartender.ai
Team members	1. Jake Atlas 2. Daniel Halteh 3. Joshua Khazanov 4. Safia Khouja 5. Kristian Nikolov 6. Greesham Simon

Problem Statement

This project serves to implement some of the methods discussed in our deep learning course, specifically those related to recurrent neural networks. Here, we specifically investigate the creation of a computer program that is capable of suggesting ingredient combinations that could be mixed in proper proportions (left to the user to experiment and determine) to create tasty cocktails. Liquor is an interesting substance in that it is truly poison, yet humans love to consume it. Creating a cocktail is a fantastic way to take poison - which in many of its raw forms does not taste good at all - and mix it with other ingredients that successfully balance the flavors and make a drink that is alcoholic but enjoyable to drink. This report provides details regarding our successful creation of a bidirectional LSTM that solves the age-old problem: how can I make a decent new cocktail? Information will be presented in an order that - in consideration of the chronology of a machine learning project - is intuitive: First, we provide information about the dataset used and the necessary preprocessing steps that must be taken before modeling can be performed. We subsequently discuss the modeling approach before presenting results. Lastly, we will present some conclusions about this project and discuss some future steps that can be taken to further improve upon the insights developed here. For information specific to the project as it relates to the Northwestern MS in Analytics Deep Learning course, see the Appendix.

Dataset

The data that was used for this project came from Kaggle and can be found here. In this dataset, there are over 500 cocktail recipes represented in tabular format, with over 41 columns of information, primarily ingredients and their amounts. The first step, beyond downloading the data, is to reshape this data so that it no longer has unnecessary information and is not presented in a tabular format. Recurrent neural networks, particularly those discussed during this course, take raw text input, so it is important to parse the data and reshape it accordingly. Otherwise, modeling would be very difficult. Below, we can see the format of the processed data:

```
['coconut rum, amaretto, orange juice, grenadine'],
['light rum, ginger beer, lemon peel'],
['vodka, peach schnapps, orange juice, cranberry juice'],
```

Some important changes have been made, notably:

1. Data has been made consistently lowercase
2. Punctuation is removed
3. Amounts and measurements are removed

It is important to note that, not only is each row in this data a cocktail, but each cocktail begins with liquor. Subsequent ingredients may contain liquor, but the first ingredient is guaranteed to be alcoholic. This is crucial because it is what enables the model to better learn that each drink must contain liquor.

However, although data in this form is ready for modeling, it is not sufficient because there are relatively few cocktails; while 500 drinks might seem like a lot to most people, a recurrent neural network needs significantly more data to produce valuable output. In order to remedy this important shortcoming of the data, we take advantage of the fact that each of the first ingredients is alcoholic. Specifically, for every cocktail, we keep the first ingredient as the first ingredient, but create new rows of data with many possible permutations of the remaining ingredients. The increase in sample size not only helps to enforce the alcohol requirement, but also greatly increases the coherency of the final model output.

Referring to the first example above: {coconut rum, amaretto, orange juice, grenadine}, we see that this permutation process increases the size of the data quite a bit. The one record is now 6 records:

1. {coconut rum, amaretto, orange juice, grenadine}
2. {coconut rum, amaretto, grenadine, orange juice}
3. {coconut rum, orange juice, amaretto, grenadine}
4. {coconut rum, orange juice, grenadine, amaretto}
5. {coconut rum, grenadine, orange juice, amaretto}

6. {coconut rum, grenadine, amaretto, orange juice}

With the data in this format, the process of modeling becomes significantly less daunting. The model will be exposed to many more combinations of ingredients that go well together.

It would be very interesting to see whether there are particular cocktails that are easy for the model to grasp or, alternatively, ones that are quite difficult, as the existence of such records might indicate that the dataset could be more carefully curated to produce an even better model, although perhaps more specialized. However, assessments of “hard to predict” or “easy to predict” records do not really apply here, as it makes more sense to view the entire corpus of cocktails as a single record from which the model learns.

Technical Approach

Each cocktail in our data is ordered—in particular, the first ingredient in every recipe is alcohol and the following ingredients include mixers, add-ins, and toppings. Adding the spirit first is a practice that conforms to typical bartender guidelines, so we applied a LSTM (Long Short Term Memory) network to encourage learning and create output in that same format.

We began by creating a minimum viable product using a basic LSTM model. This model had a single layer of size 128 and no additional modifications. The LSTM architecture, which focuses on learning order dependence, provided us with a method of developing new cocktails that maintain the order in which ingredients are added. The typical LSTM unit consists of a cell, an output gate, an input gate, and a forget gate, the latter two of which impact how the input transitions through and affects the output. After experimenting with a combination of layer and step sizes, we found the LSTM model with a single hidden layer of 128 memory cells using a step size of 2 to be our best performing model, likely due to the smaller size of our dataset. Although we began by toying with a different RNN (Recurrent Neural Network) architecture, we ultimately decided to use an LSTM network because it does not suffer from vanishing and exploding gradients.

We tried 4 unidirectional LSTMs with different layer/node combinations, and 1 bidirectional LSTM, which we decided to add after toying with bidirectional LSTMs in our second homework assignment. To compare the different models, we measured

cross-entropy loss as an initial view of how our model was performing in this semi-supervised learning approach, along with “accuracy” (defined as containing real words, containing alcohol and starting with alcohol), novelty and diversity, which were more inline with our ultimate objective - to create coherent and novel cocktails. Out of the 5 models, the bidirectional LSTM performed best in both cross-entropy loss and our real-world output metrics.

Unlike typical LSTMs, Bidirectional LSTMs basically duplicate the first recurrent layer in the network, with the first layer remaining the same and second being a reversed copy of the original input sequence.This provides the network with additional context behind our cocktail recipes and their respective sequences. One common downside to Bidirectional LSTMs is that they have a high computational cost due to having both a forward and backward pass, which, as a result, can slow down the learning process. Given the small nature of our dataset, however, we felt like the added context from the backward pass would outweigh that slight, potential slow-down. Concretely, the added context in this case helped produce more coherent results, and also reduced the amount of duplicate ingredients in a generated cocktail (we did end up cleaning the output, but it was still an interesting byproduct).

A crucial step in our cocktail generation process was cleaning the output from our LSTM model to fit our expectations and needs. We began by removing all ingredients that did not appear in the original dataset, which helped us clear the output of gibberish and poorly generated text. Although we considered retaining generated recipes that did not contain alcohol for minors or sober-people, we ultimately decided to remove non-alcoholic drinks since bartender.ai was trained solely on alcoholic drinks. After cleaning, the model, which generated a 10,000 character output, had produced 158 new drinks!

We also compared each generated recipe in the cleaned dataset with all the recipes in the original dataset to calculate the maximum jaccard similarity between each generated recipe and the originals. This post-processing step helped us grasp how accurately the model generated cocktails. For instance, we noticed that several cocktails in the generated set directly matched the cocktails in the original dataset.

We would like to consider alternate approaches to this problem that we could apply to our dataset in the future. We would like to try convolutional neural networks (CNN) or generative adversarial networks (GAM) with a starting ingredient, such as the base alcohol. These alternatives could potentially perform better because they would designate a starting ingredient instead of relying on ingredient mixing order. When

mixing cocktails, the order is significant in the more complex cocktails (adding the liquor before the egg white is crucial), but non significant in other more simple cases (vodka-cranberry will taste the same as cranberry-vodka). As a result, using an LSTM which learns order dependence might not work for cocktails that don't necessarily have a crucial sequence. In the future, our team hopes to compare our best LSTM model to these alternative models.

Results

The output of each model the team generated was assessed for accuracy, novelty, diversity, and taste. We considered these metrics and the cross entropy loss curve to determine that the bidirectional 2-layer 128-node LSTM worked the best. The metrics for this model will now be discussed (results for all models can be seen in Appendix A).

Accuracy:

The percent of cocktails in the raw output that contained no fake ingredients is 70%. The percent containing alcohol is 82%, and the percent having alcohol as the first ingredient is 71%. From a technical standpoint, a 70% accuracy is not great, however, cleaning the output is not difficult and having 82% of the output containing alcohol is strong. Our model did a decent job of identifying which ingredients are alcoholic.

Novelty and Diversity:

For every drink in the filtered output (fake ingredients removed from drinks, gibberish and nonalcoholic drinks removed from output), the jaccard similarity to every drink in the original dataset was computed, and the maximum value was recorded for each output drink. We will refer to this as maximum jaccard similarity, or MJS. The percent of novel drinks in the processed output (i.e. MJS not equal to 1) is 95%. The average MJS across all output drinks is 0.48 and 55% of the output has a MJS under 0.5. We are very pleased with the novelty of our output. Almost all of our drinks are brand new, and a majority of our dataset has a MJS closer to 0 than 1. Furthermore, The average pairwise jaccard similarity in the filtered output is 0.037, and 99% of the output is unique. Hence, our output is very diverse.

Taste:

Six bartenders were asked to vote on which cocktails (in the filtered output) they believed were tasty. 66% of the filtered output received at least 1 vote, 37% received at least 2 votes, and 20% received at least 3 votes. This is very encouraging.

The final project improved upon the MVP in every accuracy submetric. In the technical approach section, we referred to the MVP as a 1-layer 128-node LSTM using a dataset where the first drink was alcoholic. Here, we will refer to the model iteration that immediately preceded this, which was a 1-layer 128-node LSTM using a dataset with scrambled ingredients (i.e. first was not necessarily alcoholic). The MVP generated output of which 49% contained only real ingredients and 44% was alcoholic, compared to 70% and 82%, respectively, for the best model. The MVP was not assessed for novelty, diversity, or taste because the accuracy was so poor.

There may be a modified version of our problem that could be more readily solved. For instance, generating novel, diverse, and yummy cocktails is tricky, but if someone wanted to use our model to generate any drink (non-alcoholic included and also taste doesn't matter), then they could easily use our model to achieve that. They could train for a few more epochs to bump the accuracy (specifically, the % of output that contains only real ingredients) even higher. However, we don't believe this is a use case worth pursuing. Also, since the MVP was our most simple neural network experimented with architecturally, it was able to train much faster than all the subsequent models we tried. In all other aspects, the more complex models, especially the bidirectional LSTM, performed much better.

The data augmentation we conducted, which is explained in the Dataset section, improved our model performance in that we increased the training data size more than 5 fold which allowed the models to be exposed to several different combinations of ingredients.

In Appendix B, you can see the cross entropy loss curves after training each model for 60 epochs. All the models that we've tried converged at this point, and it can be seen that the bidirectional LSTM resulted in the lowest loss. In order to compare the output model, we generated a sample output of 10,000 characters using the same seed for each model, and it can be seen in Appendix B that the bidirectional LSTM (2 layers 128 nodes) and the single layer unidirectional (256 nodes) created output with the most "real" ingredients (i.e. ingredients that are in the original dataset). Benefits of ensuring alcohol is the first ingredient in the training data also ensures that the output would also create more alcoholic cocktails.

After cleaning the output cocktails by removing ingredients that did not exist, we compared the novelty and diversity of each drink by calculating the maximum Jaccard similarity between the generated cocktails with the cocktails in the original dataset. After inspecting this, the bidirectional LSTM model tested resulted in the more novel and diverse cocktails. Since the bidirectional LSTM had the lowest cross entropy loss, one of the higher performances in generating alcoholic drinks with less gibberish (nonexistent ingredients in the original dataset), and finally the higher novelty and diversity scores, we determined that this model was the best out of all the models tried.

Finally, we took the output of this best model and we had 6 bartenders inspect the output to determine if the drinks generated could possibly be any good. At least 3 bartenders voted that 20% of the cleaned output could possibly be good. We deemed this as successful since these were novel cocktails. Ultimately we will have to experiment and see if the drinks are any good.

Concluding remarks and future work

The issue with generating cocktails is that there are not many recipes with which to work and train models. As such, our training set - even after augmentation via permutations - was quite small, resulting in cocktails that were passable but not excellent. When we started the project, we did not expect that our results would even be viable cocktails, and were surprised to see that our final model not only produced coherent words, but was also able to - in some cases - simulate actual human logic, by suggesting combinations of ingredients that go well together (like rum and coke) but were not present in the original data (at least, not on their own as a standalone drink). In its current form, our model is definitely not “shovel-ready,” in the sense that using it over and over would not provide anything new and useful, as our generator would at some point start spitting out the same cocktails.

No other model we can think of can even come close to reproducing similarly coherent results, so the extra computation required to train our model was definitely worth it. That being said, our project was specifically chosen to be a good match for deep learning; in most of our internships and future jobs at the level we are starting, the problems we will be tackling are likely to be unnecessarily overcomplicated by using any deep learning model. There are many applications for deep learning outside of academia, but they are specific to a few key industries; in general, faster and simpler models will provide high enough accuracy and be easier to productionalize. The serious efforts pytorch and tensorflow have been putting into developing their products have yielded great progress in putting models that were once very niche into the hands of non-senior level

professionals, similar to the way in which other products are currently putting more basic models into the hands of entirely non-technical analysts. This suggests that we are on a trajectory towards a future where it is very possible that deep learning might be much more easy to use than it is today, with a wider range of applications. It would be interesting to see how advancements in the field of deep learning will enable improvements to projects like this one, which will no doubt become more viable with the passage of time.

For our project, as is often the case in deep learning specific applications, it was much more useful to get coherent results than it was to get explainable results (we don't actually care how the text was generated as long as it was reasonable). There are other cases where accuracy is much more valuable than interpretability, where deep learning could be more useful than a simple model; however, there are just as many cases where interpretability is key, where deep learning models would be unusable due to their nature. Increased interpretability in deep learning models would likely make it even more popular and help lead to further improvements in creating an AI bot that develops its own cocktails.

References

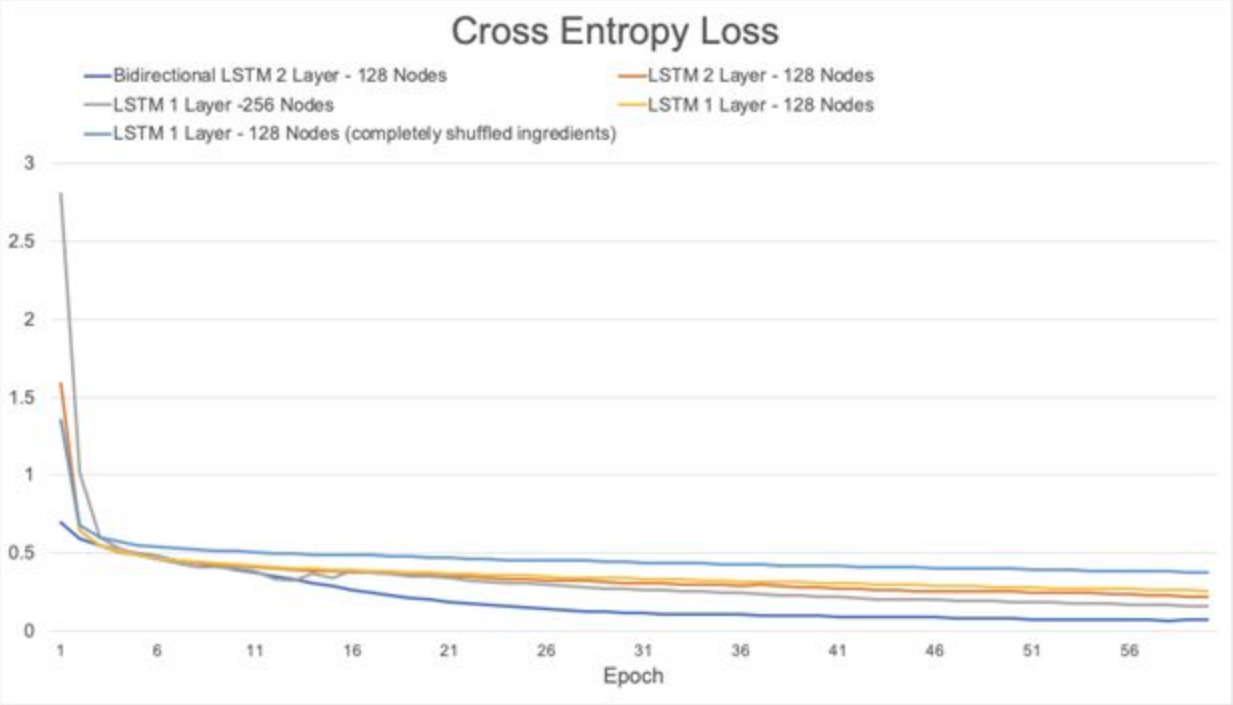
- Dataset Sources:
    - https://www.kaggle.com/ai-first/cocktail-ingredients
    - https://www.kaggle.com/jenlooper/mr-boston-cocktail-dataset
    - https://www.kaggle.com/shuyangli94/cocktails-hotaling-co/version/2
- Code Sources:
    - https://github.com/keras-team/keras/blob/master/examples/lstm_text_generation.py

Appendix

Appendix A: Model accuracy, novelty, diversity, and taste metrics

Metric		Accuracy			Novelty			Diversity		Taste (based on 6 bartenders' expertise)		
Denominator		Raw Output			Filtered Output		N/A	Filtered Output	N/A	Filtered Output		
Submetric		% contains only real ingredients	% is alcoholic	% alcohol is first ingredient	% is novel	% MJS < 0.5	Mean MJS	% is unique	Mean pairwise Jaccard	% at least 1 vote tasty	% at least 2 votes tasty	% at least 3 votes tasty
Model	1-layer 128-node (scrambled ingredients)	49	44	51								
	1-layer 128-node	63	81	69	96	53	0.48	98	0.035			
	1-layer 256-node	78	92	73	87	38	0.56	97	0.036			
	2-layer 128-node	55	88	73	92	46	0.51	97	0.042			
	2-layer 128-node (bidirectional)	70	82	71	95	55	0.48	99	0.037	66	37	20

Appendix B: Cross entropy loss plot



Comments on the Project

This project was useful in helping us to learn about some of the nuances of generative models. Having an example like this was instrumental in gaining an understanding of why a bidirectional layer is helpful in particular circumstances like this one. We also got to learn a lot about the proper ways to preprocess data for use in recurrent neural networks, which was arguably the greatest contributor to our success, even beyond that of the chosen neural architecture itself.

The schedule was adequate for completing this project, though taking steps that would require self-teaching beyond the scope of the class was generally deterred by our busy schedules and the unusual structure of remote school. The second homework (alongside the COVID-19 generative sample code) was instrumental in providing an understanding of the RNN basics we would need. It was difficult to start on the project until we had done this, but luckily, we were not hindered by software or computing limitations. By using Google Colab, we were able to avoid competing for Deepdish use. The second homework assignment definitely helped prepare for this project as described in the Technical Approach section. We took the LSTM and modified the input and the architecture to better suit the problem we were trying to solve.

AI and deep learning are very much important aspects of the data science and machine learning curriculum. It is definitely important that we cover this in the MSiA curriculum. Although deep learning seems somewhat niche, as simpler, faster models can handle many data science problems with more interpretability, it has progressively become a household name as it permeates through many more application use cases than ever before. Interpretability is also becoming more important as machine learning solutions are being applied to more industries than ever before. One improvement to this course that could be useful is possibly learning how to make deep learning models more accessible and interpretable. Essentially, one improvement would be to learn how one would get key stakeholders in a business who have no knowledge of the inner workings of a deep neural network to buy in.