Making chicken recipes: But it’s a surprise every time.

Ashim Dahal

Abstract

This paper includes study of three different models approaches, one of which might help an imaginary online cookery show or a YouTube content creator to create content related to cooking chicken also while never running out of ideas. The result from the paper can also be used to experiment in the kitchen but if you don’t have a broad idea of cooking then leave it to the kitchen experts and lets just focus on the recipe generation.

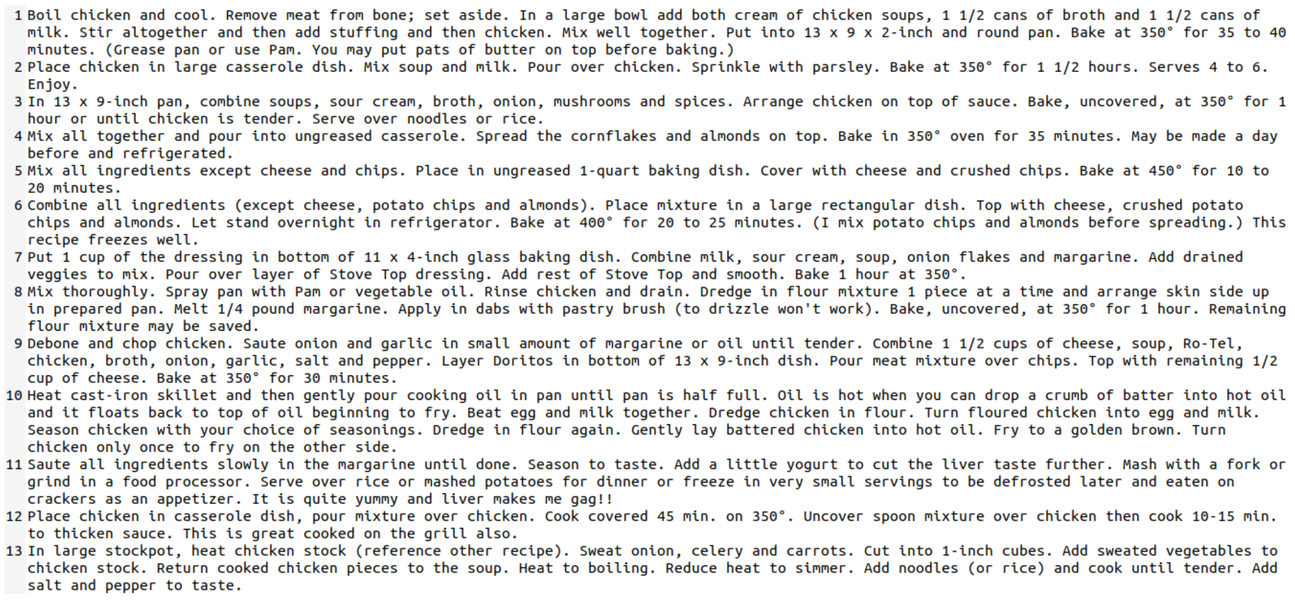
1. INTRODUCTION

Using RNNs [1] to generate texts have been one of the wide applications of deep learning in the past few decades and after this paper there is one more way to have fun with RNNs [1] while also doing research. As stated by Karpathy in his article [2] Recurrent Neural Networks(RNNs) are unreasonably good at what they are supposed to do i.e, perform exceptionally well on sequential data but as many believe that its GRUs [3] and LSTMs [4] that have brought limelight and foreground to sequential problems and also omitted the exploding gradient problem that simpler RNNs had to suffer from.

While there are generative adversarial networks like MASKGAN [5] such highly powerful and accurate model cannot be applied to this rather simple task because the model is based on input sequence being incomplete and the model being fed somewhat the beginning of the sentence the recipe generator of this particular need needs to be completely random and without any prior inputs. The input can be either a word that would start the chain of recipe generation or should be a zero vector.

2. BACKGROUND TO THE DATASET

When it comes to recipe generation the first dataset to come in mind would be RecipeNLG (cooking recipes dataset) [6]. But as I wanted to make this whole model fairly simple and not heavy computationally and the dataset RecipeNLG (cooking recipes dataset) was fairly large I choose to synthesize my own chicken recipe dataset which would be small and from online sources. For the simplicity of things the dataset is made so that it only contains the descriptions or directions of cooking procedure and does not contain the conditional part, the conditional part being referred to the available ingrident to the content creator but this should not be the case to worry about for the online cookery show or generating daily content for an blog post. The recipe for the dataset is imported from the results on chicken keyword search on the online cooking recipe site cookbook. The dataset consists of 150 different and unique chicken recipes from different origins and with varying length of words with some number and characters complexity which needed to be removed in the code. The text file consists of 150 lines, each with collection of sentences that make up a small recipe to be generated.



```
Boil chicken and cool. Remove meat from bone; set aside. In a large bowl add both cream of chicken soups, 1 1/2 cans of broth and 1 1/2 cans of milk. Stir altogether and then add stuffing and then chicken. Mix well together. Put into 13 x 9 x 2-inch and round pan. Bake at 350° for 35 to 40 minutes. (Grease pan or use Pam. You may put pats of butter on top before baking.)
Place chicken in large casserole dish. Mix soup and milk. Pour over chicken. Sprinkle with parsley. Bake at 350° for 1 1/2 hours. Serves 4 to 6. Enjoy.
In 13 x 9-inch pan, combine soups, sour cream, broth, onion, mushrooms and spices. Arrange chicken on top of sauce. Bake, uncovered, at 350° for 1 hour or until chicken is tender. Serve over noodles or rice.
Mix all together and pour into ungreased casserole. Spread the cornflakes and almonds on top. Bake in 350° oven for 35 minutes. May be made a day before and refrigerated.
Mix all ingredients except cheese and chips. Place in ungreased 1-quart baking dish. Cover with cheese and crushed chips. Bake at 450° for 10 to 20 minutes.
Combine all ingredients (except cheese, potato chips and almonds). Place mixture in a large rectangular dish. Top with cheese, crushed potato chips and almonds. Let stand overnight in refrigerator. Bake at 400° for 20 to 25 minutes. (I mix potato chips and almonds before spreading.) This recipe freezes well.
Put 1 cup of the dressing in bottom of 11 x 4-inch glass baking dish. Combine milk, sour cream, soup, onion flakes and margarine. Add drained veggies to mix. Pour over layer of Stove Top dressing. Add rest of Stove Top and smooth. Bake 1 hour at 350°.
Mix thoroughly. Spray pan with Pam or vegetable oil. Rinse chicken and drain. Dredge in flour mixture 1 piece at a time and arrange skin side up in prepared pan. Melt 1/4 pound margarine. Apply in dabs with pastry brush (to drizzle won't work). Bake, uncovered, at 350° for 1 hour. Remaining flour mixture may be saved.
Debone and chop chicken. Saute onion and garlic in small amount of margarine or oil until tender. Combine 1 1/2 cups of cheese, soup, Ro-Tel, chicken, broth, onion, garlic, salt and pepper. Layer Doritos in bottom of 13 x 9-inch dish. Pour meat mixture over chips. Top with remaining 1/2 cup of cheese. Bake at 350° for 30 minutes.
Heat cast-iron skillet and then gently pour cooking oil in pan until pan is half full. Oil is hot when you can drop a crumb of batter into hot oil and it floats back to top of oil beginning to fry. Beat egg and milk together. Dredge chicken in flour. Turn floured chicken into egg and milk. Season chicken with your choice of seasonings. Dredge in flour again. Gently lay battered chicken into hot oil. Fry to a golden brown. Turn chicken only once to fry on the other side.
Saute all ingredients slowly in the margarine until done. Season to taste. Add a little yogurt to cut the liver taste further. Mash with a fork or grind in a food processor. Serve over rice or mashed potatoes for dinner or freeze in very small servings to be defrosted later and eaten on crackers as an appetizer. It is quite yummy and liver makes me gag!!
Place chicken in casserole dish, pour mixture over chicken. Cook covered 45 min. on 350°. Uncover spoon mixture over chicken then cook 10-15 min. to thicken sauce. This is great cooked on the grill also.
In large stockpot, heat chicken stock (reference other recipe). Sweat onion, celery and carrots. Cut into 1-inch cubes. Add sweated vegetables to chicken stock. Return cooked chicken pieces to the soup. Heat to boiling. Reduce heat to simmer. Add noodles (or rice) and cook until tender. Add salt and pepper to taste.
```

Fig 1 : recipes on dataset.

2.1 Understanding the dataset

The dataset has 150 unique recipes from the cookbook's members home secrets, with a vocabulary of 1211 unique words, letters and characters. Adding the unknown word token <unk> to the vocabulary made it a total of 1212 unique words in the dataset. The largest or say longest data in dataset consists of 730 words. Yes I know that is a long recipe for cooking chicken. So I had three approaches to making the vectors of training data.

2.1.1 Sequencing 100 words per training example from pretrained embeddings

This may be the laziest and easiest approach to making a training example. On this approach to the training data, the entire text is treated as a single continuous line and 100 words are taken at a time for vectorization. This may already explain the problem with this approach. This approach is good for making training data to create similar vocabulary like the text generation examples on Shakespeare's vocabulary shown in tensorflow example for RNNs [1] but this approach does not have the well representation of a recipe. Lets say the first recipe is 120 words then in the training example the second example would start from the 101th word of the first recipe, this makes the training not well representative of the actual recipe but is good for teaching the model just the context of words used in the training dataset. The embedding and problems that came with it is discussed in the next approach as it become more prominent in that approach.

2.1.2 Using padding for training example with THRESH and pretrained embedding

In this approach for the training example the training dataset is padded. Each one of the recipe is fed as the training example and to better be the representative of the recipes on the entire dataset, any recipe that exceeds the threshold value of the hyper parameter THRESH is truncated and the recipes that do not meet the required threshold are padded with the <unk> token. The THRESH hyper parameter was required because the word embedding was too big and the processing was not too efficient. This made the model to better learn the words but the problem in this and the previous method was both used pretrained embeddings from the NLPL word embeddings repository [7] which contained 291186 words of the English language from the English Wikipedia Dump of February 2017 Gigaword 5th Edition corpus. The main issue was the embedding vocabulary did not have some of the words used in the dataset so the word vectors would often be gibberish and not understandable.

2.1.3 Using padding for training example without THRESH and fresh embedding

The idea was to dump the pretrained embedding layer and to train in on the model itself. This was fairly easy because the vocabulary now was 1212 and not 291186 which meant we could use higher dimension for embedding and also use more GRU units without running to OEM exceptions in tensorflow. The THRESH hyper parameter was removed and the whole training model was padded to the largest training example so the training examples were the shape of (150,730).

2.2 Training batches

Now that the training inputs were figured, the outputs were to remain the same inputs but one time step in the past. So if the first two training words were boil chicken then the first input was to be *boil* and its output should be *chicken* and so on. The first approach had training batches of 32 so each batches would be shape (32,100), this was possible without running into OEM issues in tensorflow because the the batch only had 100 word vectors in them. However in the second approach the batch size had to be deducted to 4 batch size as the embedding had huge vocabulary of 291186 and the shape was (4,THRESH). One possible remedy to this could be to only keep the words from the embeddings that were in the vocabulary but that would remove the scalability of model when new training examples were to be added. So, each time the embedding vectors were to be truncated according to the new words in the recipe vocabulary. The final approach allowed to push things to the next limit. The input were in batches of (32,730) and the embedding dimensions were too increased by 3.2 times without exhausting any resources because the embedding had only the 1212 number of vocabulary. The number of GRU units and embedding dimension increase had good positive in the model as shown by the figures in loss.

3. Training Models Architecture

Each of the three models have the following architecture with change in all of the hyper parameters.

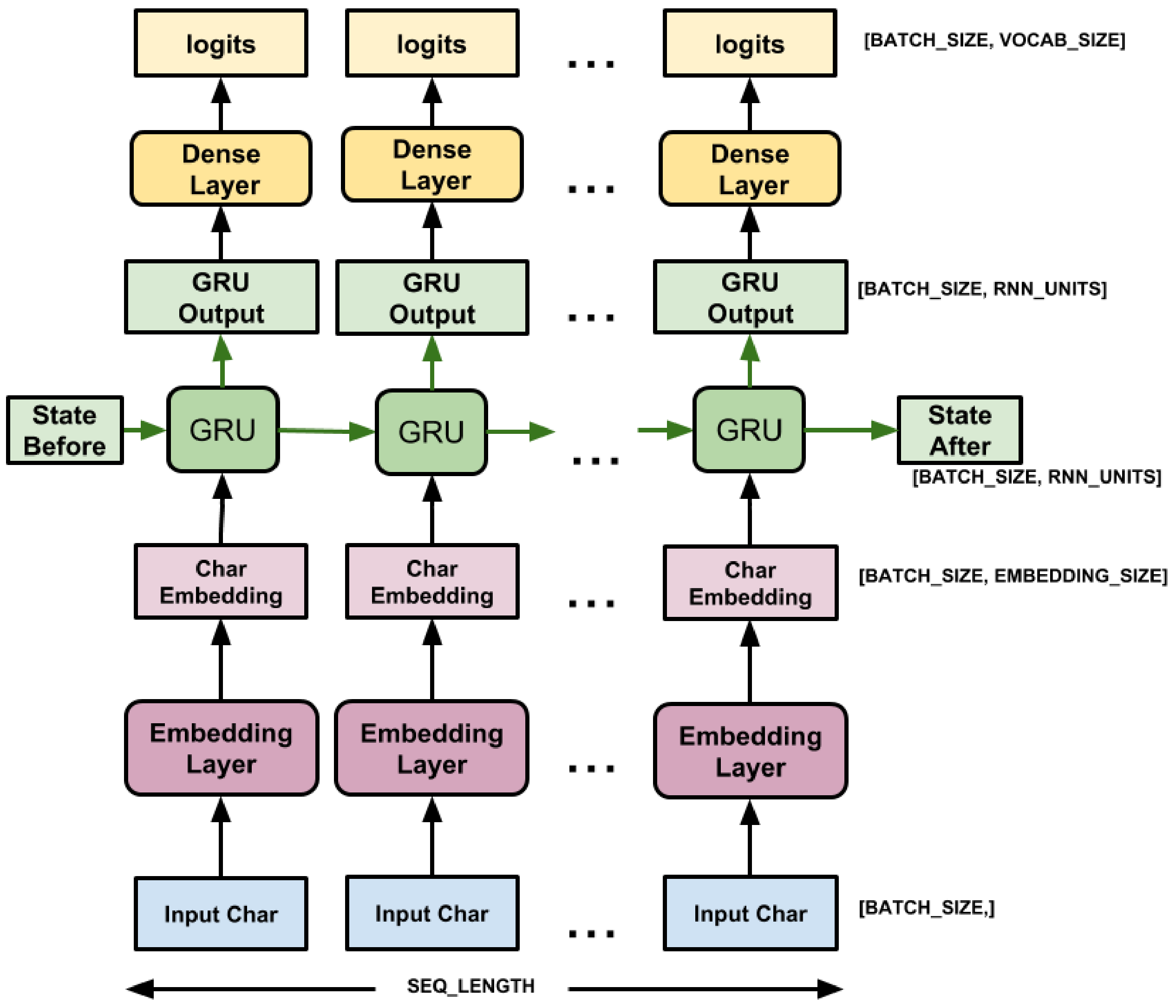


Figure 2 : pic credit : tensorflow

The number of GRU units and the embedding layer had massive effects on the models that were created. The three approaches to the models are named as follows:

3.1 InitialEasyModel

The dataset fed to this model is the first approach to the dataset as discussed in [2.1.1]. The model cannot be said to be trained exceptionally well or anything close to that but it did a fair job in training but was nowhere near the other two models. As the problem with the method of dataset discussed the model was unable to make good use of data. The number of GRU units used in this model was 1512 GRU units and it had output the logits as shown in the figure 2. The model had a loss of 5.4547 after 20 training epochs, the optimizer of choice was ADAM [8] without gradient clipping and the loss function used was sparse categorical cross entropy [9].

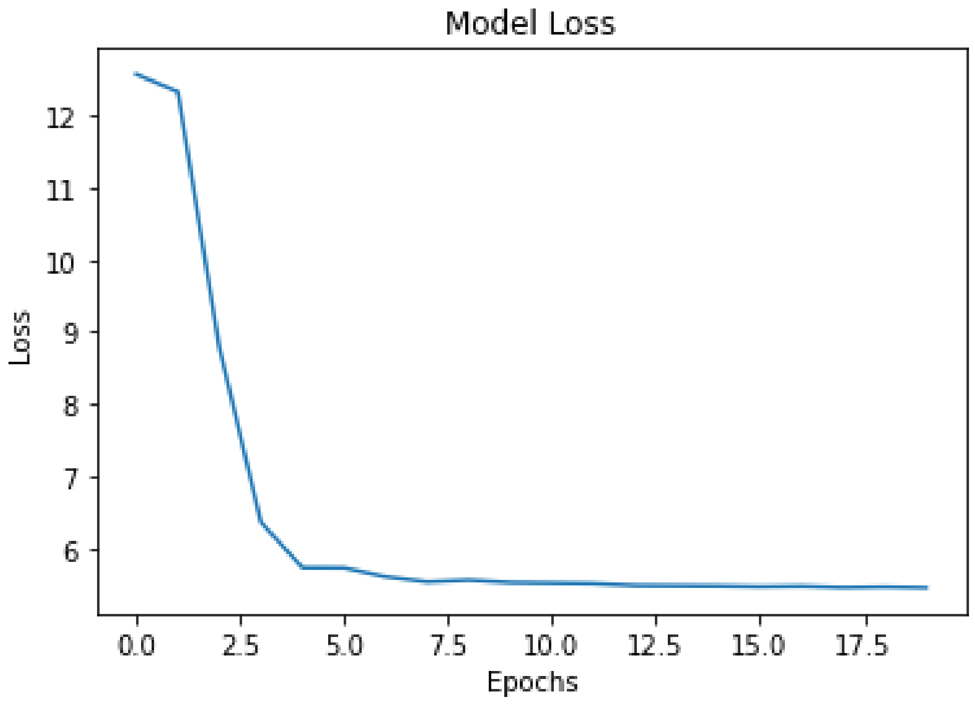


fig 3: loss vs epochs

3.2 Model with padded dataset

The dataset fed to this model is the second approach to the dataset as discussed in [2.1.2]. The model trained exceptionally well as compared to the first model with just 516 GRU units. It had output the logits as shown in the figure 2. The model had a loss of 0.6647 after 20 training epochs, that is already way better than the initial model but since the model had batch size of 4 it was on the slower size to train the optimizer of choice was ADAM [8] without gradient clipping and the loss function used was sparse categorical cross entropy [9].

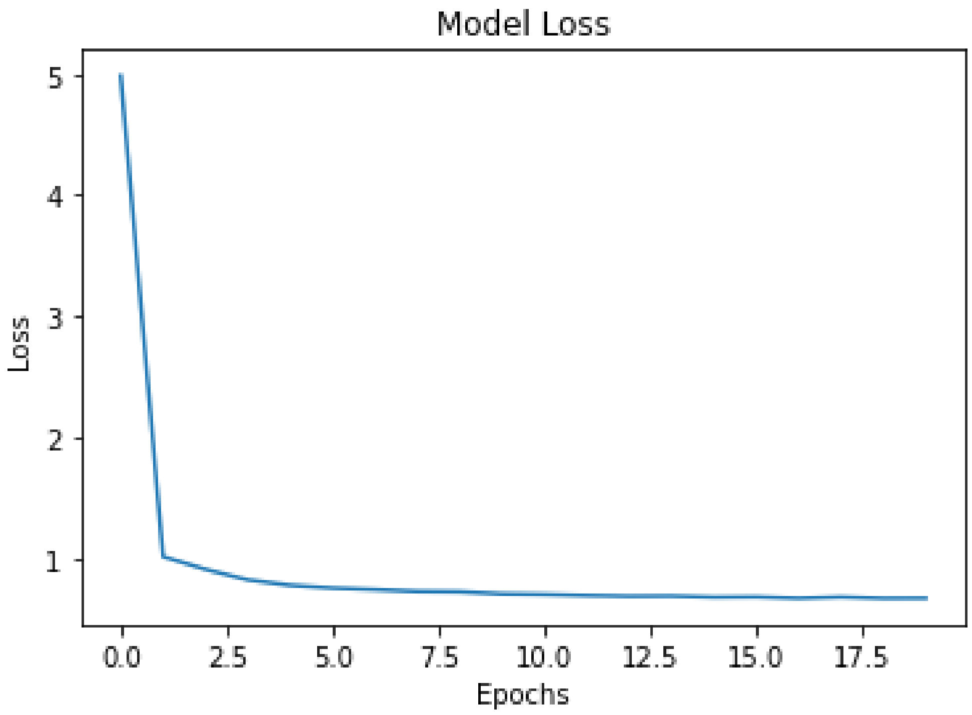


fig 4 : loss vs epochs

3.3 Model with padded dataset and larger Embedding with less vocabulary

The dataset fed to this model is the third approach to the dataset as discussed in [2.1.3]. The model trained exceptionally well as compared to the first two models with just 4024 GRU units and the embedding dimension of 1024. It had output the logits as shown in the figure 2. The model had a loss of 0.2679 after 20 training epochs, that just crushes the other two models in training and was also faster than the second one due to larger batch size the optimizer of choice was ADAM [8] without gradient clipping and the loss function used was sparse categorical cross entropy [9].

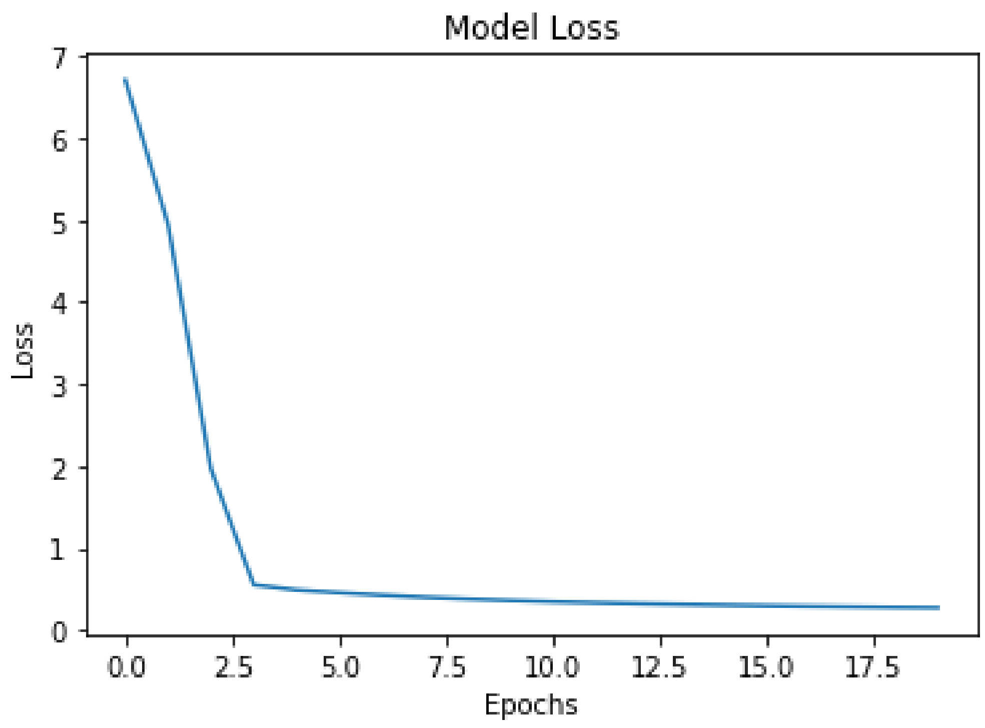


fig 5. loss vs epochs

4. Generator Model

The generator model is the standard generator model by the works of tensorflow in their notebook [10] of text generation.

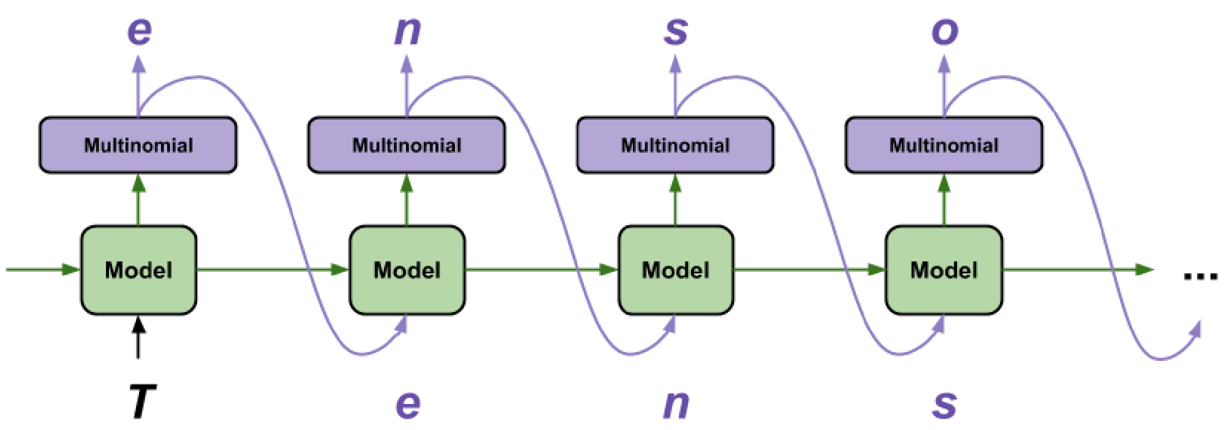


fig 7. pic credit : tensorflow

4.1 Generation by first model

As expected the first model was , well just a model, a poor performing model in the training example and did not hold up to generate much detailed instructions in the recipes. The generation of the model is as follows given the initial word ` put `:

`*put splatter hot in 20 pork in minutes add almonds oil brown ginger liver fantastic and lightly remove almonds place mayonnaise margarine onions oil spray and add together reduce 1/2 stir mushrooms dredge spray 3 in lowering and little vegetable add add 5 the mixture milk turn crushed half low constantly*`

4.2 Generation by second model

This model was slightly better than the first one but not as good as you would want to be in the objective of posting in a blog post, the length of text vary as I was tuning how long sentence might alter the generation but they do not alter much.

`*boil a casserole bake butter large 9 set and milk butter 2 curry 3 in 5 simmer breasts add a large poultry tastes waxed excess casserole melted heat hours bowl of and slow fire seasoned small constantly onion dish serve bring bake chicken dish using color of broth tender pot add and made serve and remove may 3 of pepper aside and and chicken and stir* `

4.3 Generation by third model

Although the model looked promising in the training loss part but this one is not going to cut it but at least this model has learnt somewhat of a recipe, if not it just eventually manages to make a perfect line and still has some way to go. This model is the best pick for me.

`*boil water marinate chicken cook on top and cook about 5 to the soups and chopped celery and cut chicken in saucepan melt butter yogurt top and salt and cover cornflakes into a few steamed rice avocado, basil and chicken and simmer place chicken on medium heat through serve with celery and broth lemon juice on rack in a as to the oil heat thoroughly and onion cover and garlic and broil herbs, enhances the*
`

5. Outcome and future of the experimental project

Although none of the models is absolutely perfect the third model is the best in the business for just the simple architecture of fig 2. The model can be made complicated or instead of just regular training we can train two models in styleGAN [11] fashion which might help the model to be adequate but taking into account of how easy and simple this model was to build with just few lines of codes it is fascinating to see and research how this amazing technology of deep learning can enforce our life to be better and easier. We as humans can do much more in much less time when we have the power of such technologies and as Andrew ng states *machine learning is the new electricity* and the electronics are very intriguing. As of text generation this model can have few extra things in its near future which would make it a better generator. First of the thing would be more data, in many parts of deep learning it is fairly true that the more of data we have the more powerful our model becomes, this one however had only 150 custom made data which is not going to cut it. So, addition of data would surely be more helpful. The second thing that could be done to make this model more accurate would be more and more of data cleaning. The regex in the data is also one of the factors that is hampering the outcomes, as I only made sure to remove the periods on the data it was not beneficial for it.

The key finding of this experimental project could be that more amount of data is always handy. Word level and character level of text generation would be better if we decide to make the dataset as the vocabulary, for more complicated tasks like question answering and named entity recognition the pretrained embedding might be useful for making relations with unseen new words but in this case, all we want to do is try to replicate the vocabulary and the sequence of words which does not require pretrained embeddings. The other finding of the experimental project is that it will be handy to truncate the pretrained embeddings if we don't want to enforce new words into the recipe, this would tremendously help to not exhaust our resources.

6. References

[1] Rumelhart, David E; Hinton, Geoffrey E, and Williams, Ronald J (Sept. 1985). "Learning internal representations by error propagation." Tech. rep. ICS 8504. San Diego, California: Institute for Cognitive Science, University of California.

[2] Andrej karpathy "The Unreasonable Effectiveness of Recurrent Neural Networks"

[3] *Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”*

[4] *Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory"*

[5] *William Fedus, Ian Goodfellow, Andrew M. Dai “MaskGAN: Better Text Generation via Filling in the______”*

[6] *Michał Bien, Michał Gilski, Martyna Maciejewska, Wojciech Taisner,Dawid Wisniewski , Agnieszka Ławrynowicz “RecipeNLG: A Cooking Recipes Dataset for Semi-Structured Text Generation”*

[7] *http://vectors.nlpl.eu/repository/*

[8] *Diederik P. Kingma, Jimmy Ba Adam: “A Method for Stochastic Optimization”*

[9] *Panagiotis Meletis and Gijs Dubbelman “On Boosting Semantic Street Scene Segmentation with Weak Supervision”*

[10] *https://www.tensorflow.org/text/tutorials/text_generation*

[11] *Tero Karras, Samuli Laine, Timo Aila “A Style-Based Generator Architecture for Generative Adversarial Networks”*