RNN Chef

Paul Galloway, Anastasiya Lazareva, Sharath Rao
University of California, Santa Barbara

***Abstract*—The goal of this project was to train a Recurrent Neural Network (LSTM) to generate food recipes based on input images. A dataset containing 45,000 food images and accompanying cooking instruction was collected. Image features were extracted using the Inception 3 Convolutional Neural Network and used as input into an LSTM model along with training recipes. Hyperparameters were tuned and the best performing model was trained for 30 epochs. Validation and test results were evaluated using the BLEU score. The resulting performance indicate that the generated recipes improve in their accuracy with the actual ones, as the epochs increase. Selected recipes can be viewed on the RNN Chef website [4].**

I. Introduction

The problem of automated recipe generation on food images is somewhat similar to the well-known image captioning problem. Recipe generation is different from image captioning in that the recipes are much longer (700 to 900 words, including characters for newline and tabs) than typical captions (10 to 20 words) and requires more long-term dependency information when being generated by an RNN. A possible advantage of recipe generation over image captioning is that the problem domain is restricted to food-related items and activities and therefore one can expect a smaller variation in image features as well as a smaller caption vocabulary size compared to traditional image captioning tasks.

One way to perform image captioning using a combination of image and sentence description features is to train a Recurrent Neural Network that can infer the correspondence of image features to their descriptions as in [1]. It is based on learning the inter-modal correspondences between language and visual data. However, the state of the art approach is to use an LSTM with attention[5]. In this approach attention is provided to different parts of the image and a natural language description of the image is learned. Our architecture is based on this approach [5]

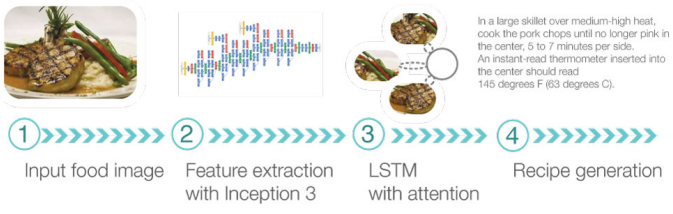

Fig. 1. Conceptual Model Diagram

II. Methods

A. Pre Processing

Data, including food images, ingredients, and cooking instructions was collected from allrecipes.com. We developed a python scraper in order to scrape the images and text data. We used the pattern-web library [6] in order to do this.

One of the observations we had during our pre-processing stage was that in the original Show, Attend, and Tell caption pre-processing stage, sentences were tokenized using blank spaces. This approach caused our vocabulary to be around 70,000 words. On closer observation we found some patterns like ”water;” ”water:” ”water,” ”water.” and ”1/2” ”1/”. This was because we did not consider punctuation during string splitting.

After fixing issues such as the one described above (by creating tokens for punctuation), we were able to reduce the size of vocabulary to just 10,000 words.

B. Transfer Learning

Image features were extracted using Inception V3, a winning ImageNet competition architecture [2]. It allows one to extract the high level features of an image dataset without extracting any low level over-fitted features. The output of this was a vector of 2048 features for every image. This is the input to the LSTM model described here.

C. PCA

Our initial training data was 35000 x 2048. This made it very difficult to fit into the GPU during training. Therefore we performed Principal Component Analysis using the library available in [7]. We were able to reduce the 2048 features down to just 512 features for every image.

D. LSTM Model

Image features along with the corresponding recipes were used as inputs to the ”LSTM with attention” model implemented in Tensorflow [8]. The Show, Attend, and Tell model was released in April of 2016 and has achieved state-of-the-art performance in image captioning tasks. The model is trained to focus attention on important image features in order to generate the corresponding image description in the caption.

Our LSTM model was not pre-trained. Initialization of the weights was done randomly. We supported 512 hidden units initially, due to memory constraints. This was primarily a problem when attempting to train on the GPU. We could not fit the entire recipe representation in the memory of the GPU. We had to settle for just the first 100-200 words in the recipe.

E. Training in the Cloud

We trained multiple models using Amazon EC2 cloud service. We will only discuss the two best models we used here.

Our first model was trained using a c4.4xlarge cluster. It had 30GB memory and a 16 core processor. We set the hidden units to 512 initially. Training a model for one epoch took about 3.5 hours. We could go up to 20 epochs for this model.

The second model had a similar configuration. One major difference was that we used a refined vocabulary to train this model. The total vocabulary size reduced from 70000 to 10000. This model was trained using a g2.2xlarge GPU instance. The GPU available on this VM had 4GB of memory.

The following hyper-parameter values were used.

Maximum length of sentence = 150/200 (actual=900)
Learning rate : 0.01
Training data size = 35000 images
Hidden Units = 256, 512, 1024, 2048
Epochs = 5, 10, 15, 20, 25, 30 (final model)

For evaluation, we used the BLEU score. The BLEU score is a metric frequently used to evaluate the quality of machine translation. We used it for both validation and testing purposes.

F. Tuning Hyper-parameters

We trained several models by increasing the hidden units from 256 to 2048. Our observation was that BLEU score was increasing very, very slowly. However, we couldn't play around with the learning rate very much due to limited computational resources.

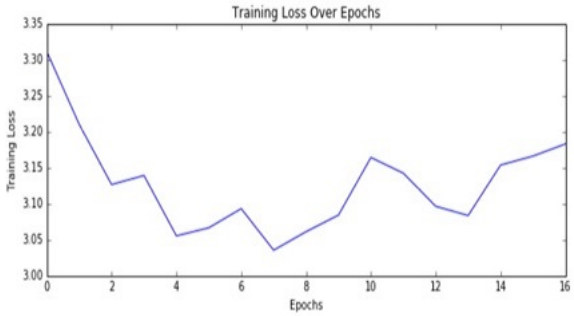

Fig. 2. Graph showing Training Loss over epochs for Model 2

During training we observe that the loss decreases as the epochs increase up to Epoch 7. After that the behaviour is sporadic. Our understanding is that the learning rate was on the high side.

III. Results

We achieved a BLEU score of 6-8% for Model 2 on the validation and test set. It clearly shows that our model had to trained for higher epochs. Typically, image captioning models are trained for 1000 epochs. The high turnaround time for this number of epochs was a significant obstacle given our more limited computational resources

Two example recipes are shown below. We show for two separate models the evolution of the resulting recipes. Asterisks signify phrases that were repeated by the RNN successfully.

Model 1: One Skillet Corned Beef Hash:

Actual Recipe:

2 tablespoons vegetable oil
2 onions, chopped
4 potatoes, peeled and chopped
2 (12 ounce) cans corned beef
1 tablespoon *ground black pepper*
5 tablespoons cider vinegar

Heat oil in a large skillet over medium high heat. Saute onions and potatoes until slightly browned, then stir in corned beef. Season with pepper and add vinegar 1 tablespoon at a time, cooking for 3 to 5 minutes in between each addition. Partially cover skillet, reduce heat to medium-low and cook, stirring occasionally, for about 20 minutes or until potatoes are tender

Evolution of the output recipe (// means that anything afterwards is repeated)

For Epoch 6:

2 cups white sugar 1 cup chopped fresh // and the chicken

For Epoch 10:

1 tablespoon chopped fresh and the chicken // and cook until the chicken

For Epoch 15:

1 teaspoon ground black pepper to a boil. cook and stir in the preheated oven until the center // and cook until the center

For Epoch 20:

2 cups white sugar 1/2 teaspoon *ground black pepper* to a boil. cook and stir in the chicken and the chicken is tender, about 10 minutes. add the chicken and the chicken is tender, about 10 minutes. // add the chicken and the chicken is tender, about 10 minutes.

Model 1 was only trained for 20 epochs, but as can be seen, it begins to make progress towards learning a basic element of the recipe. Model 2 (shown below) makes further progress after just 10 more epochs

Model 2: Herbed Cream Cheese-Stuffed Lamb Burgers

Actual Recipe:

4 ounces Philadelphia Cream Cheese, softened
1 tablespoon chopped fresh chives
1 tablespoon *chopped fresh parsley*
1 1/2 pounds ground lamb
1 tablespoon Worcestershire sauce
1 teaspoon salt
1/2 teaspoon *pepper*
1 tablespoon olive oil
4 onion sandwich rolls, split, toasted
1 bunch watercress, thick stems removed, separated into equal portions
Mix cream cheese and herbs until well blended. Roll into 4 balls; flatten each into disk on sheet of waxed paper. Refrigerate 15 min. or until firm.
Combine meat, Worcestershire sauce, salt and pepper. Shape into 8 thin patties. Place cream
cheese disks on 4 patties; cover with remaining patties. Press edges together to seal.
Heat oil in large skillet on medium heat. Add patties; cook about 5-6 min. on each side or until done
(160 degrees F). Serve in rolls topped with watercress.

Epoch 15: // 1 / 2 cup butter

Epoch 25: // 1 / 2 cup butte

Epoch 30: the and *chopped fresh parsley*, // 1 / 2 teaspoon ground black pepper

For this recipe with 10 more epochs the model was able to pick out two distinct features of the recipe instead of just 1 as in the previous model.

IV. Conclusion

Clearly there is plenty of room for improvement with the model demonstrated here. At the very least it shows that the implementation of Show, Attend, and Tell cannot easily be used out-of-the-box for recipe generation. More care probably needs to be taken in the initialization of the model. One way to try this is to use a pre-trained LSTM model. But aside from the particular LSTM implementation used, there is plenty of room for improvement in how the data is fed into the RNN. One way to improve the BLEU score for recipe generation could be by providing additional information for the LSTM to learn. We could separate out the instructions from ingredients and try to predict one given the other. Further up the pipeline one could consider trying to find a better source of images. Perhaps a site that has images in a more standard arrangement. This could be, for example, all images being shot in front of some monochromatic background and at a similar angle. Obviously, such options are somewhat limited when you are dealing with crowd-sourced data.

TABLE I. Division of Labor

Anastasiya	**Paul**	**Sharath**
Scraper code	GPU Experimentation	Preparing dataset
Inception feature extraction	Tuning Parameters	Running PCA
Tokenization	Cloud Preparation	Recipe Data Analysis
Evaluation metric		Tuning parameters
Website		Cloud preparation,Training and Testing

References

[1] Vinyals, Oriol, et al. *Show and tell: A neural image caption generator*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[2] Szegedy, Christian, et al. *Rethinking the Inception Architecture for Computer Vision*, arXiv preprint arXiv:1512.00567 (2015).

[3] Ilya Sutskever et al. *Generating Text with Recurrent Neural Networks*, Proceedings of the 28th International Conference on Machine Learning, WA, USA, 2011

[4] RNN Chef, http://www.alazareva.github.io

[5] Kelvin Xu, Yoshua Bengio et. al *Show, Attend and Tell: Neural Image Caption Generation with Visual Attention*, Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015

[6] Pattern Web in python, http://www.clips.ua.ac.be/pages/pattern-web

[7] PCA in scikit learn, http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[8] Show Tell Attend Library in Git, https://github.com/jazzsaxmafia/show$_attend_and_tell.tensorflow$