CPSC 8810: Deep Learning – Homework 2

by - Abhijeet Amle

***Abstract*—In this paper, I have explained the implementation of sequence to sequence model for creating video captions.**

I. Introduction

In this assignment, for video captioning I have chosen a sequence to sequence LSTM model with Attention mechanism. For further optimizing the model, I have used gradient clipping. The model has achieved a **BLEU score of 0.6815**

II. Structure

I have created 4 files for this task – input_preprocessing.py, model.py, train_model.py, and test_model.py. Brief description of each file as below:

1. input_preprocessing.py

This file reads all the training video features and the corresponding training video captions and stores it in a feature dictionary and a caption dictionary. Dictionary contains all the captions including multiple captions for each video id. As the captions are English words, this dictionary has further converted to vectors by keeping the minimum occurrence of a word count to 3.
Vocabulary size: minimum count >= 3

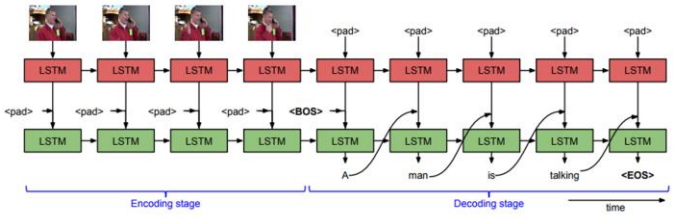
This will not store all the words in the dictionary, but it will store the most used ones. I have stored these dictionaries' pickle objects so that it can be used directly during the training of the model. Also, have stored word to index mapping and index to word mapping as pickle objects.

Appended the below keywords to each caption before storing it in the dictionary:
<pad> - padding to make all sentences of equal length (padding will be applied before training)
<bos> - beginning of a sentence
<eos> - ending of a sentence
<unk> - unknown for the dictionary (or word cannot store to dictionary)

The advantage of storing objects is we don't have to read the captions and video features for training set multiple times. During training, we can directly read these pickle object files and train our model.

2. model.py

Implemented a sequence to sequence encoder-decoder model using 2-layer LSTM with 1024 units in each layer for both encoder and decoder.



(2-layer LSTM structure)

Hyperparameters used for LSTM layers in both encoder and decoder:
Forget bias = 0.7
Dropout = 0.5 (for training)
Dropout = 1.0 (for testing)

Added a Bahdanau-Attention on encoder outputs and an Attention-Wrapper on the decoder LSTM cell and Attention mechanism object. This attention mechanism will assign a weight to each input word which is then used by the decoder to predict the next word in the sentence.

To prevent the neural network from exploding gradients, I have implemented gradient clipping. This will further optimize the training of the network.

Hyperparameters for creating model:

Learning rate: 0.0001
Optimizer: Adam
Maximum gradient norm = 5.0
Maximum encoder steps=64
Maximum decoder steps=15

3. train_model.py

This file reads the earlier saved pickle objects of dictionaries. Then have padded all the sequences of captions to have equal length of sequences. Created a batch of video features and captions of batch size 55 for training the model, have trained

the model for 200 epochs. Training loss for 1st epoch was around 2.95 and the same for 200th epoch was around 0.613.

Parameters used for training:

Batch size=55
Training sample size=1450
Total epoch=200
Video features dimension=4096
Video frame dimension=80

```
Epoch 190/200, loss: 0.596917, Elapsed time: 18.99s
Epoch 191/200, loss: 0.616280, Elapsed time: 16.80s
Epoch 192/200, loss: 0.673485, Elapsed time: 18.36s
Epoch 193/200, loss: 0.603085, Elapsed time: 17.21s
Epoch 194/200, loss: 0.680922, Elapsed time: 18.74s
Epoch 195/200, loss: 0.618698, Elapsed time: 17.59s
Epoch 196/200, loss: 0.661000, Elapsed time: 20.02s
Epoch 197/200, loss: 0.599525, Elapsed time: 17.00s
Epoch 198/200, loss: 0.612878, Elapsed time: 14.91s
Epoch 199/200, loss: 0.634631, Elapsed time: 20.14s
Epoch 200/200, loss: 0.613085, Elapsed time: 18.85s
Total training time: 45.09m
```

(Screenshot of model trained for 200 epochs; total time is recorded for 150 epochs as first 50 epochs were already executed) Total training time for 200 epochs is around 60 minutes on Colab with GPU enabled.

4. test_model.py

For training the model, I have loaded the index to word and word to index dictionary objects and the pretrained model.

Created a dictionary of testing features having a **batch size of 1** for evaluation of the model. This testing of the model will generate an output file which we can test against a BLEU which is a metric for evaluating a generated sentence to a reference sentence. With these hyperparameters configured, this model has achieved a BLEU score of 0.6815. Below is the screenshot of BLEU score.

```
[ ]  1 ! python3 /content/bleu_eval.py '/content/output_testset.txt'

     Average bleu score is 0.6815228578208792
```

References

[1] https://vsubhashini.github.io/s2vt.html
[2] https://www.tensorflow.org/tutorials/text/nmt_with_attention
[3] https://medium.com/@dhirensk/tensorflow-addons-seq2seq-example-using-attention-and-beam-search-9f463b58bc6b
[4] https://colab.research.google.com/github/innarid/seq2seq/blob/master/seq2seq.ipynb
[5] https://github.com/tensorflow/addons/blob/master/tensorflow_addons/seq2seq
[6] https://www.tensorflow.org/addons/api_docs/python/tfa/seq2seq/AttentionWrapper
[7] https://www.programcreek.com/python/example/90312/tensorflow.clip_by_global_norm
[8] https://blog.floydhub.com/attention-mechanism/