CPSC 8430: Deep Learning

HW 2 - Report

M Sabbir Salek

GitHub Link: https://github.com/msabbirsalek/DL-with-PyTorch/hw2

Introduction

In this homework, we developed a sequence-to-sequence (S2VT) model to generate captions for given video clips. The objective of this is to provide a video clip as the input while the developed model predicts what is happening in the video and generates the output as a string of caption.

Requirements

- Python 3.9.7
- Pytorch 1.10.2
- Scipy 1.7.3
- Pandas 1.4.1
- MSVD datatset (1450 videos were used for training and 100 videos were used for testing)

Dictionary

First, we loaded the labeled file and created a dictionary for vocabularies. The tokens presented below were used to create the dictionary,

- <PAD> is used for padding the sentences to the same length
- <BOS> denotes the beginning of a sentence
- <EOS> denotes the end of a sentence
- <UNK> is used when the word is not found in the dictionary or to just ignore an unknown word

Base Model

The baseline for the seq2seq (S2VT) model consists of two layers of Recurrent Neural Networks (RNNs). The first layer is responsible for processing and encoding the videos and it is named as the “encoderRNN” class in the python script. The second layer is responsible for decoding and generating the output and is named as the “decoderRNN” class. “decoderRNN” is written to segment the captions using tokens depending on the beginning and ending verses of a sentence and to process based on the video to generate the actual words as the output. The figure below

depicts the process of encoding and decoding using the “encoderRNN” and the “decoderRNN” classes, respectively.

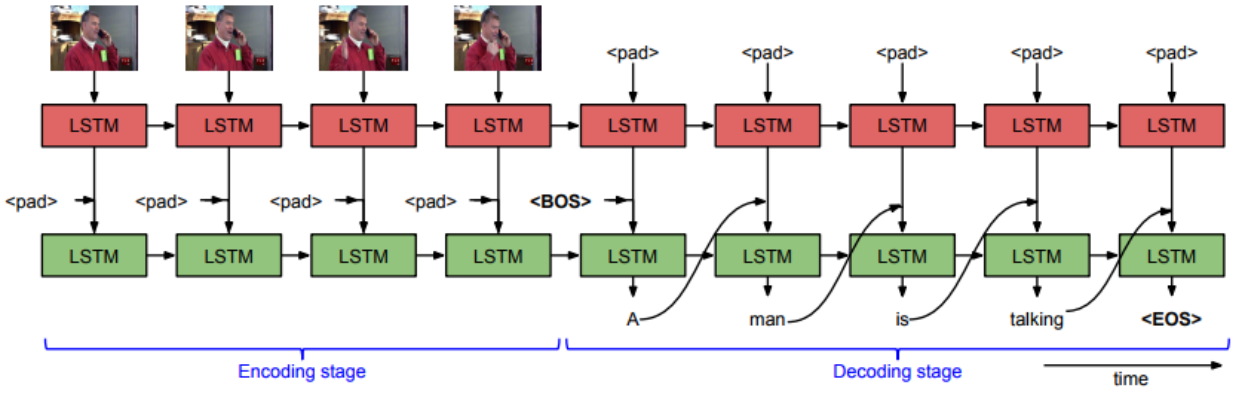

Fig. 1 S2VT Model

Attention Layer

On the encoder hidden states, we implemented an attention layer in order to get better performance from the base model. We adopted the structure for this attention layer from Shen and Huang (2016). The hidden state of the decoder and the output of the encoder are used as a matching function to get a scalar, which passes through the softmax layer and then the last hidden state is sent to the following time step of the decoder.

Schedule Sampling

For inference, the model-generated token replaces its previous unknown token which can cause errors getting accumulated over the generated sequence. This problem is known as the exposure bias problem. To solve this problem, we used the ground truth as the input for the RNNs while training the model. This process is depicted in the figure below.

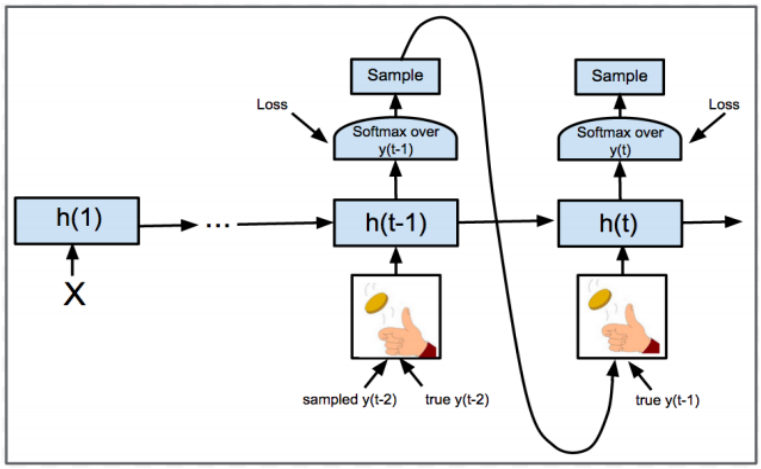

Fig. 2 Schedule sampling.

In this project, the CrossEntropyLoss is used as the loss function and at the final stage of training, the bleu_eval is used to determine the BLEU score of the model. We have trained 20 models for varied batch sizes, hidden layer sizes, dropout percentages and word dimensions. Top 5 models were trained for 10 epochs and the results are shown in the table below. Based on the highest BLEU score we selected the 3rd model on the table as the best model and trained it for 50 epochs to get the final model.

Model #	Learning rate	Batch size	Hidden layer size	Dropout percentage	Minimum count of vocab size	BLEU score
1	0.001	16	256	0.2	3	0.600094
2	0.001	16	512	0.2	3	0.690617
3	0.001	32	128	0.2	3	0.699342
4	0.001	32	128	0.3	3	0.67757
5	0.001	32	128	0.4	3	0.666792

Reference

Shen, Y., & Huang, X. J. (2016, December). Attention-based convolutional neural network for semantic relation extraction. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (pp. 2526-2536).