Jean-Michel Tine

CPSC-8430

Deep Learning HW-2

Git-Hub Link:

Introduction

The task at hand is to create a sequence-to-sequence (S2VT) model capable of generating captions for video clips. The goal is to input a video clip into the model and have it predict what is happening in the video, outputting a descriptive caption in the form of a string.

Requirements

- Python 3.9.7
- Pytorch 1.10.2
- SciPy 1.7.3
- Pandas 1.4.1
- Spicy 1.8.0

Dataset

MSVD dataset (1450 videos were used for training and 100 videos were used for testing)

Dictionary

To begin, we loaded the labeled file and constructed a vocabulary dictionary using the tokens listed below:

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

Model

The S2VT model's baseline consists of two RNN layers. The first layer, known as the "encoderRNN" class in the python script, encodes and processes the input video. The second layer, called the "decoderRNN" class, is responsible for decoding and generating the output caption, segmented

into sentences using tokens based on their beginnings and endings. The "decoderRNN" processes the video to generate the actual words of the output. The figure below illustrates the encoding and decoding process using the "encoderRNN" and "decoderRNN" classes, respectively.

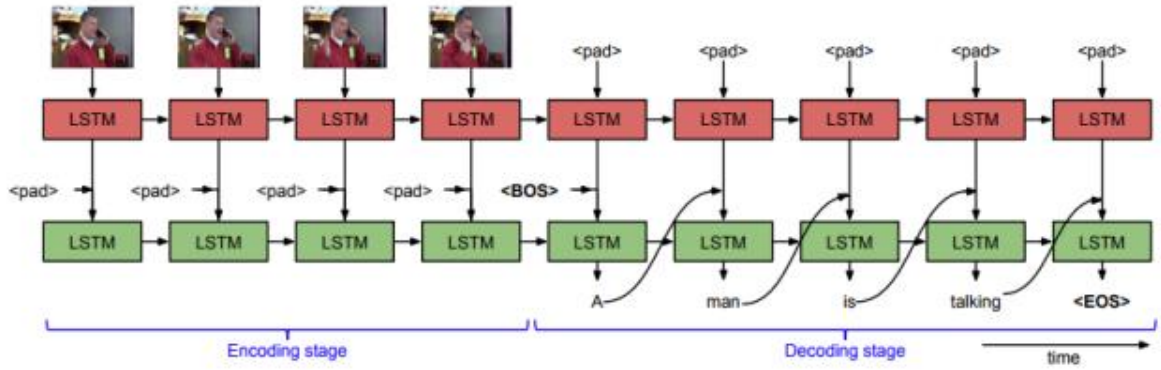

Figure 1: S2VT Model

Attention Layer

To improve the performance of the base model, we added an attention layer on the encoder hidden states. This attention layer structure was inspired by Shen and Huang's work from 2016. The decoder's hidden state and the encoder's output are used as a matching function to generate a scalar, which is then passed through the softmax layer. The resulting output is used to derive the last hidden state, which is sent to the next time step of the decoder.

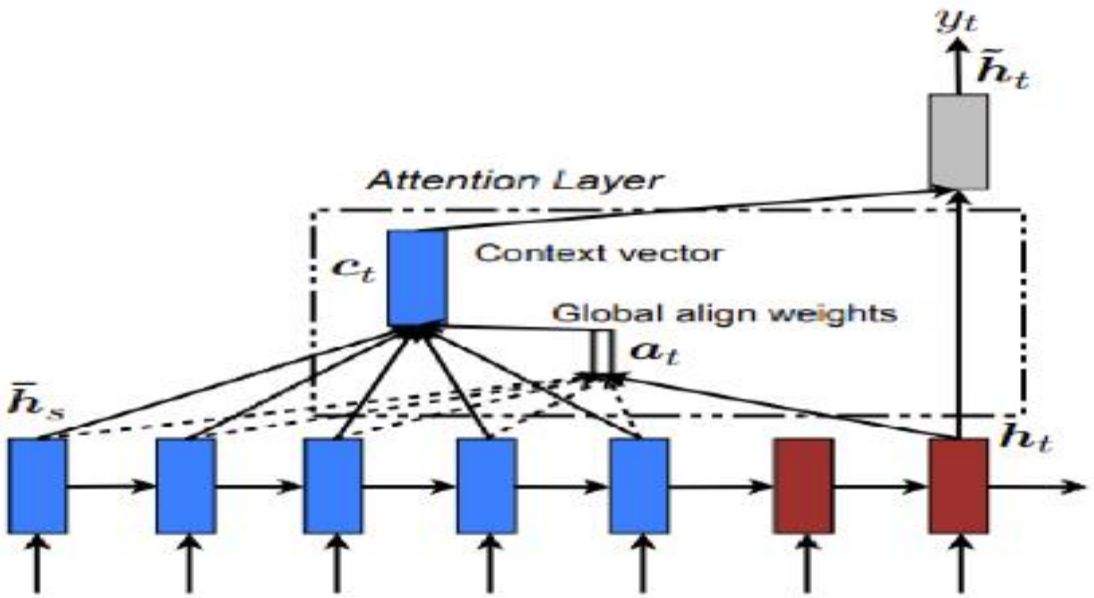

Figure 2: Attention layer

Schedule Sampling

During inference, replacing the previous unknown token with the model-generated token can result in accumulated errors over the generated sequence, which is referred to as the exposure bias problem. To overcome this issue, we trained the model using ground truth as the input for the RNNs. The figure below illustrates this process.

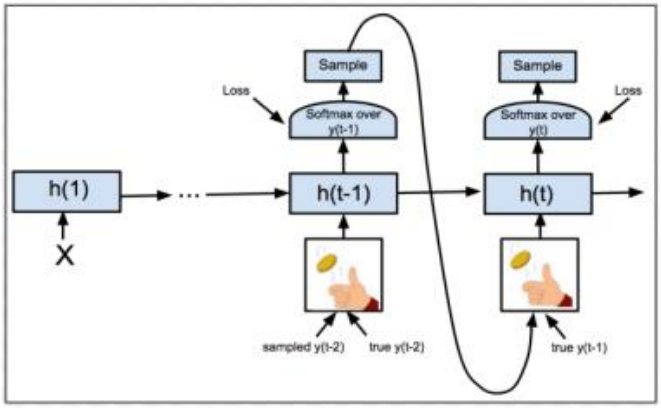

Figure 3: Schedule sampling

In this project, we utilized the CrossEntropyLoss function as our loss function. During the final stage of training, we evaluated the model's performance using the BLEU score. We trained 20 models with varying batch sizes, hidden layer sizes, dropout percentages, and word dimensions. The top 5 models were trained for 10 epochs and their results were organized in a table. The model with the highest BLEU score was selected, and we trained it for an additional 50 epochs to obtain the final model.

Loss Function: CrossEntropyLoss

BLEU score of the model: bleu_eval

Minimum count of vocabulary size for all models is 3

Model	Learning rate	Batch size	Hidden layer	Droup out Percentage	BLEU score
1	0.001	16	256	0.2	0.600094
2	0.001	16	512	0.2	0.690617
3	0.001	32	128	0.2	0.699342
4	0.001	32	128	0.3	0.67757
5	0.001	32	128	0.4	0.666792