Greeshma Chandra Shekar C20084146 greeshc@g.clemson.edu

CPSC-8430
Deep Learning
Homework 2
Video Caption Generation

GitHub link: https://github.com/Greeshma-C20084146/hw2.git

Problem Statement

Use a sequential-to-sequential model to create a video caption for the input video. The code will take a video as input and produce a stream of captions that describe the actions in the video. Recurrent neural networks enable the aforementioned.

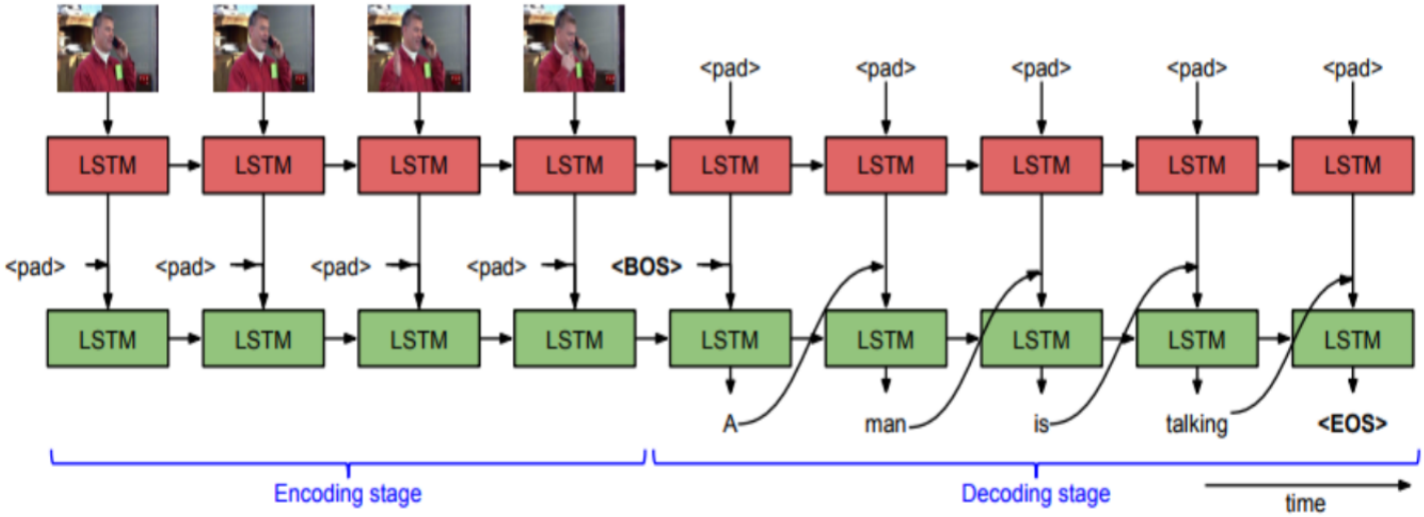
Loading and Data Preprocessing

1) Loading and preprocessing the data.
   - def getFeatures - Filename: Provides the features from a JSON file.
   - def getLabels - Index: Based on the feature's index from a JSON file, provides you with the Label for that feature.
   - def getIndicesFromSentence - Sentence: Provides an array that maps sentences' words to their corresponding indexes.
   - def getSentenceFromIndices - Index: Provides you with an array of indexes that are converted to sentence words.
   - def sample_minibatch(): Mini-batches are set as the batch size by this function.
2) Creating 2 dictionaries:
   - Word_to_count_dict: Translates all training labels' terms into an index.
   - word_to_index_dict: Mapping every index to a word.
3) A dictionary, as described in the slides, is a necessary component of the seq2seq concept. A simple word-to-phrase mapping is shown below.
   - PAD: Lengthen the sentence by the same amount.
   - BOS: Sentence beginning; indication to produce output sentence.
   - EOS: End of sentence, end of output sentence.
   - UNK: Token representing an undefined word in the dictionary.
4) The input data cannot be sent to the model directly. The sentences must be divided into words, and those words must then be represented numerically before being combined into a single-hot vector. The dictionaries have distinct indexes for the words.
5) Spaces and other special characters are removed from the input to clean it up. After that, the sentence is normalized to lowercase.

Greeshma Chandra Shekar C20084146 greeshc@g.clemson.edu

Network Architecture

1) Generating the graph
   - Short video - Pre-processed video frames: Input Placeholder X.
   - Caption that depicts the video: Output Placeholder Y.
   - Padding the input: shape= [batch_size, hidden_units]
   - Word embedding: shape= [vocab_size, embedding_size (that is number of hidden_units)]
   - Developing LSTM cells with 2 layers and 128 hidden components.



2) Best Test Bleu Score is ≈ 0.76

Model Used	
Training Epochs	500
LSTM Dimensions	128
Batch Size	128
Learning Rate	0.001
Optimizer Used	Adam Optimizer
Vocab Size	Min count > 3

The entire model is a two-layer LSTM structure, which is internally divided into an encoding stage and a decoding stage.

Encoding Stage

- Pre-processed video frames with a size of [Batch Size, 4096] every timestep for 80 frames are the input for the LSTM1.
- The embedding vector of the special word, <pad> is passed with the output of LSTM1 to LSTM2 as the input.

Greeshma Chandra Shekar C20084146 greeshc@g.clemson.edu

Decoding stage

- A zero-tensor with a size of (batch_size,4096) every time step is the input for the LSTM1.
- The output of LSTM1 at that time step and the output of LSTM2 at the previous time step are combined to form the input to LSTM2.
- Every timestep, the LSTM2 produces a tensor of size (batch size, hidden units). This result is then multiplied by the word embedding matrix's transpose to yield the logit, of size, (batch_size, vocab_size).
- The final output sentence is created by appending the argmax along the logit's columns to a tensor variable. This will be a tensor of size (batch_size, sequence_ length) at the conclusion of the sequence generation.
- A lower-dimension representation called the embedding matrix is used to look for the word embedding of the output word. The LSTM2 input will receive this at the following timestep.
- The Softmax Cross-Entropy loss of the logits with respect to the labels at that time step is used to determine the loss per time step. The overall loss per batch is calculated by adding up all of these losses across the length of the output sequence.
- Adam optimizer is utilized to reduce the aforementioned loss.

Model training

The model runs for 500 epochs, but a good validation score is achieved within these iterations after it converges. The current bleu test score is generated every 50 iterations. The best bleu score produced so far on the test dataset is logged, and the model parameters are saved each time a better score is produced. This is a rapacious method of maximizing the BLEU score on the test dataset and it may not be the best approach, but it worked for me.

Model testing

Checking the model output and the BLEU score is the first step in the evaluation process. Each pair of sentences will be fed into the model, which will then provide anticipated words. The correct index to the value will then be determined using the argmax of the values.