Cpsc 8430 - DEEP LEARNING
Homework-2
Harshini Gaddam

Github link: https://github.com/Harshini-Gaddam/Cpsc-8430-HW2_Harshini

Introduction:

Captions for the input video must be generated in this assignment.
A short video that is of 10-20 seconds is provided as the input and the corresponding caption for the video has to be generated as the output.

To generate captions, a sequence-to-sequence model with an attention mechanism is used to create a model, train, and test.

Environment requirements:

MSVD (Microsoft Video description) dataset that has 1450 video snippets for training, 100 videos for testing are used to generate the captions. Number of frames are kept constant during training for accurate results.

Palmetto cluster is used to run the codes that are used in this assignment.
Adam Optimizer is used and the learning rate used is 0.0001.
Loss function: Cross-entropy loss
bleu_eval is used to calculate the bleu score.
Various batch sizes like 10, 20 are used.

The flow that is followed to generate the captions for the video input is
Data preprocessor, Base model (S2VT-Sequence to Sequence with video to text), Encoder and decoder, Attention layer, Training and testing, BLEU score.

Process involved in S2VT:

The S2VT model uses a convolutional neural network (CNN) to encode a series of video frames, which results in a fixed-length feature vector. The encoder-decoder architecture, which consists of two Recurrent Neural Network (RNN) models: an encoder and a decoder, is then fed this feature vector.
The decoder RNN receives the feature vector after the encoder RNN has processed it to create a fixed-length representation of the video content. By predicting the following word in the sequence based on the preceding words and the encoded video material, the decoder RNN creates the output word sequence one at a time. This procedure is repeated until an end-of-sequence token is produced or a maximum sequence length is achieved.

Combining maximum likelihood estimation and beam search decoding, the S2VT model is trained. Cross-entropy loss is used to optimize the model during training to reduce the discrepancy between the projected output sequence and the actual sequence. Using beam search decoding, which includes examining the space of potential output sequences and choosing the one with the highest probability, the model generates the most probable word sequence during inference.

The model, train includes data preprocessor, base model(S2VT), training. The model, layers include encoder and decoder layer, attention layer. Testing code tests the model and outputs bleu score. Seq2seq.sh shellcode executes the testing and outputs txt file. It takes feat from the testing data which is already given and outputs the txt file in our desired location.

1.1) Data preprocessor:

This function reads the features and captions from training videos and stores them in feature and caption dictionaries. In data preprocessing process, numericalization or vectorization is done to map words.

Dictionary: As the sentences from input data cannot be directly fed into the model, these sentences are broken into words and numerical indices and used as an input. Each word in the text data can be changed to its appropriate numeric value once the dictionary has been generated. Tokens are used to create words in the dictionary.

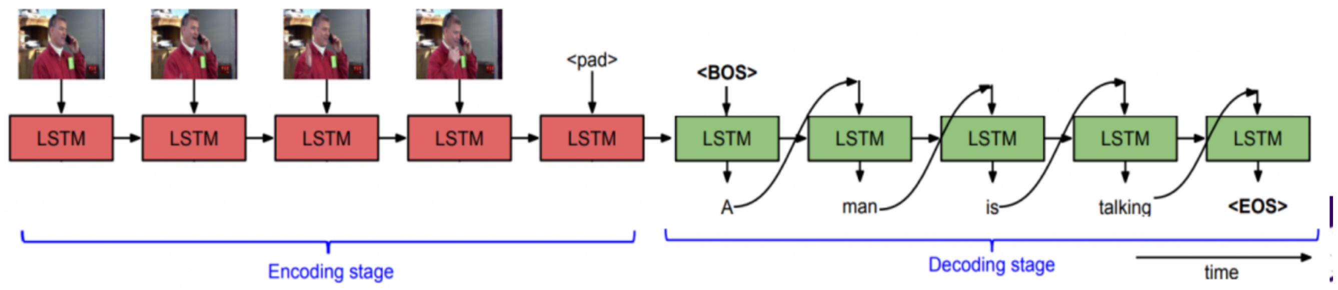
Tokens: The dictionary contains unique tokens for sentence beginning and end markers, unknown words, and padding.

<BOS>: Begin of sentence, a sign to generate the output sentence
<EOS>: End of sentence, a sign of the end of the output sentence.
<UNK>: Use this token when the word isn't in the dictionary or just ignore the unknown word.

Sequence-to-sequence model:

Two Recurrent Neural Network (RNN’s) encoder and decoder are used to process the input and generate the output respectively. The input is trained to generate the output.

Base Model flow:



Base line model is created to establish a performance standard and determine minimum level of accuracy that must be achieved by advanced models.

1.2) Encoder & Decoder:

Encoder: The input sequence that the model needs in order to construct the output sequence, is fed into the encoder, which then creates a fixed-size vector representation of the complete input sequence (context vector).

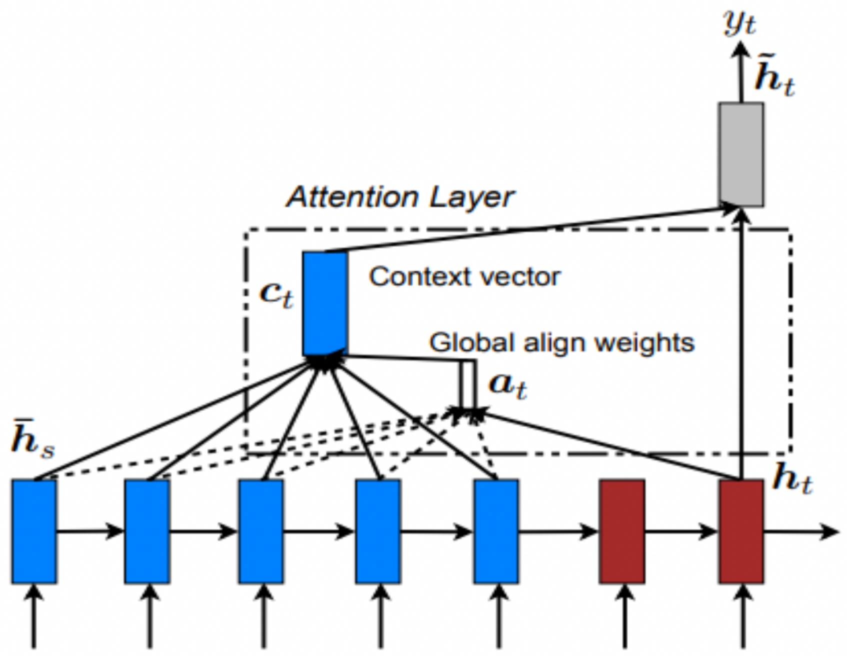
Decoder: The context vector from encoder is taken by decoder and generates output sequence. The decoder creates the subsequent output at each step by using the previous output as input.
The output sequence's length or dimensionality may differ from the input sequence's.

LSTM’s and GRU’s (Gated Recurrent Units) are used as the solution to short-term memory. They possess inbuilt devices known as gates that can control the information flow. LSTM and GRU are used to address the issue of vanishing gradients in conventional RNNs, which can make it challenging for the model to capture long-term relationships in the input data.
Compared to LSTM, GRU is simpler and computationally efficient.

1.3) Attention layer:

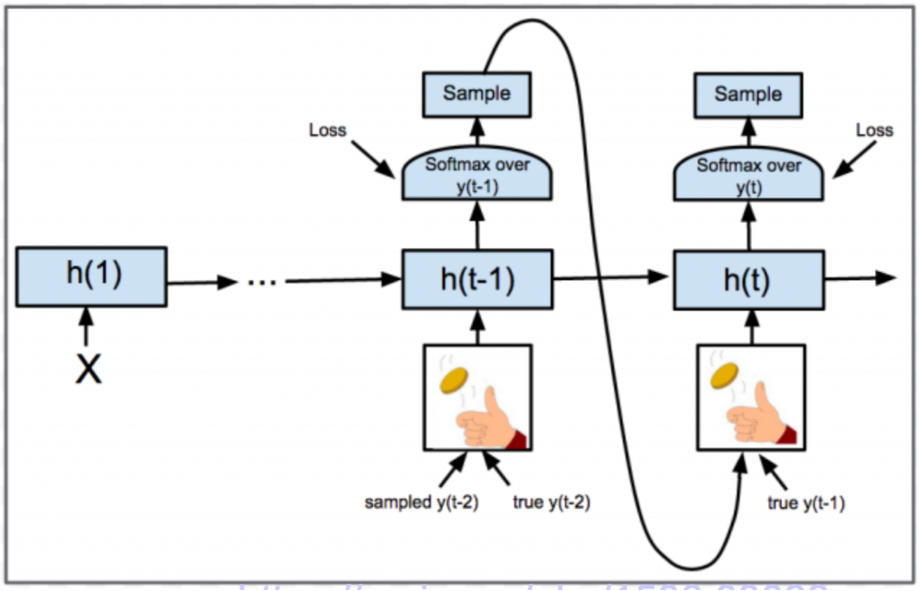
It is implemented on encoder hidden states to model long-term dependencies in sequential data more effectively. At each decoding step, attention layer allows the model to peek at several input sections.

Recurrent neural networks (RNNs) employ an attention layer to selectively concentrate on input sequence segments when creating an output. Instead of treating all inputs equally, it enables the RNN to dynamically select which portions of the input sequence to focus on. The RNN output and the entire input sequence are both inputs to the attention layer. Each input element is assigned a set of attention weights that indicate how important it is to the output at the current time step. Then, using these attention weights, a weighted sum of the input sequence is calculated, where the weights represent the significance of each input element.



Schedule Sampling:

The problem called ‘exposure bias’, that arises when RNN is trained on ground truth data but then fed its own predictions during inference replacing the ground truth. The RNN is fed the ground truth input with probability p and its own predicted output with probability 1-p at each time step during training. Normally, throughout training, the value of p is annealed, starting at a large value and gradually dropping to zero.



1.4) Training and Testing:

Using the training label data json, word count is created, and these words are indexed to form encoded sentences. Encoded sentences combines with the respective video features and serves as the ground truth.
Training model is defined in train.py and testing model is run in test.py. The test function generates the captions. The ground truth caption and the generated output caption are compared. Bleu_eval function generates a bleu score. Bleu score up to 0.7 is achieved and it fluctuates between 0.5 - 0.7, when the model is trained and tested using the provided input data.