CpSc 8430: Deep Learning Homework2

Name: Hemanth Reddy Dasari

Email: hdasari@clemson.edu

Github link: https://github.com/hemu-git-hub/Homework2.git

Abstract

Video caption generation mainly focuses on developing algorithms that can generate the illustration of the video automatically. It is done using the techniques of natural language processing and computer vision that allows the machine to grasp the video's intent and output it. The main task in video caption generation is to deal with the temporal nature of the video. So the model that we use should be sensitive to temporal nature. In this model, we have used two Long Short Term Memory (LSTM) for encoding and decoding. The MSVD (1450 videos for training and 100 videos for testing) dataset are used for assessing the model. The beam search algorithm is used for generating the captions for the video.

Technologies used:

- Python 3.9.13
- Numpy 1.23.5
- Pandas 1.5.2
- Tensorflow 2.5.0
- Cuda 11.0
- Pickle 0.7.5

Dataset Description:

The dataset that is used in this model is MSVD (Microsoft Video Description) dataset. This dataset has about 1450 videos for training and 100 videos for testing the model. The total runtime of the videos varies from 10-15 seconds. The videos in MSVD dataset are stored in MP4 video format which is the most supported file format. H.264 video codec is used for efficiently compressing videos. This dataset is downloaded from the link provided in the HW2 PowerPoint presentation.

Model Description:

The initial step is to pre-process the MSVD dataset which involves extracting the video frames and the video's natural language description and converting them into a format that syncs with Recurrent Neural Network (RNN) model. In this step, padding sequence and masking are applied to the dataset. Masking is used for handling variable-length sequences. But to perform the task, the model needs fixed length variables. The masking simply sets the values to a special marker which tells the model to just ignore during execution. This process improves the performance of the model.

The following tokens are used for pre-processing the data.

- **Data pre-process :**

  Tokens : <PAD>, <BOS>, <EOS>, <UNK>

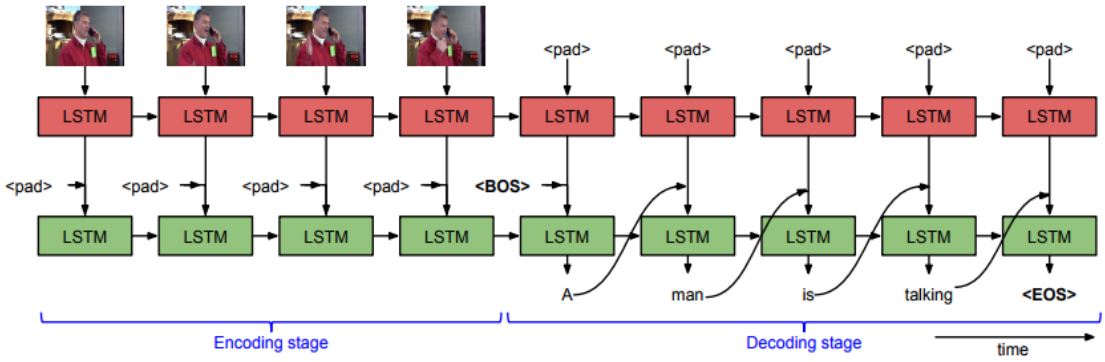
  - <PAD> : is used to add extra padding to ensure that all sentences have the same length
  - <BOS> : is used to indicate the beginning of a sentence when generating output.
  - <EOS> : is used to indicate the end of a sentence when generating output.
  - <UNK> : is used to represent words that are not in the vocabulary or to indicate that they should be ignored.

- **S2VT:**

S2VT performs the video caption generation using sequence to sequence model. The model used for video caption generation employs a stacked Long Short-Term Memory (LSTM) architecture that processes both visual and linguistic information. The visual information is extracted from the sequence of frames using Convolutional Neural Networks (CNNs) that provide outputs in the form of RGB and/or optical flow. The stacked LSTM then takes in these visual features and generates a sequence of words that describe the content of the video. By utilizing this approach, the model is able to effectively capture both the spatial and temporal information of the video and produce accurate and descriptive captions.

- **LSTM structure:**

The video caption generation model consists of two stacked Long Short-Term Memory (LSTM) layers that work together to learn a representation of a sequence of frames and decode it into a descriptive sentence. The first LSTM layer is responsible for modeling the visual features of the video sequence and is colored red. The second LSTM layer, colored green, models the language given the text input and the hidden representation of the video sequence. The model uses special tags, <BOS> for the begin-of-sentence and <EOS> for the end-of-sentence, to structure the output sequence. Additionally, zeros are used as padding when there is no input at a particular time step. Overall, this approach enables the model to accurately capture both the visual and linguistic information of the video, leading to more informative and descriptive video descriptions.



Two-layer LSTM structure

- **Attention Layer:**

The attention layer allows the decoder to selectively focus on different parts of the input sequence during the decoding process. This means that at each decoding time step, the model can examine and weigh the relevance of various sections of the input sequence, rather than relying on a fixed-length context vector. After computing the weights for each of the encoder's hidden states using the attention mechanism, the model combines them by taking a weighted sum. The decoder then uses this context vector to generate the next word in the output sequence, based on its understanding of the previous word and the context provided by the vector.

- **Schedule sampling:**

schedule sampling is a method designed to address the problem of "exposure bias" that occurs when training models. Exposure bias happens when there is a mismatch between the data that the model encounters during training (which is often the correct, or ground truth, data) and the data that it encounters during inference (which is based on its own predictions). To address this issue, schedule sampling randomly selects either the ground truth data or the model's own predictions from the previous time step as input during training, gradually shifting towards using the model's own predictions as the training progresses. Through this exposure, schedule sampling improves the accuracy and reliability of the model's prediction.

- **Beam Search:**

Beam search is a popular search algorithm that is frequently utilized in natural language processing and other associated domains for producing probable output sequences from a probabilistic model. It functions by selecting a certain number of paths, or a beam width, and then computing the probabilities of all possible subsequent steps for each path at each stage.

Results:

Using the sequence.py file, the sequence-to-sequence model was created, and using train.py this model was trained. For the training, the parameters mentioned in the HW2 presentation were used.

As the epochs are trained, simultaneously Bleu scores of the epochs are also calculated and are shown in descending order.

- The above files are run in the palmetto cluster.

- The average Bleu score at the end of the training is 0.68455.

References:

https://www.cs.utexas.edu/users/ml/papers/venugopalan.iccv15.pdf