CpSc 8430: Deep Learning Homework2

Name : Pallavi Raguri

Email: praguri@clemson.edu

Github link: https://github.com/PallaviRaguri/Homework2

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. The MSVD (Microsoft Video Description) dataset, utilized in this model, comprises approximately 1450 training videos and 100 testing videos. Each video's duration ranges between 10 to 15 seconds. These videos are primarily in the MP4 format, known for its wide compatibility. For effective video compression, the
H.264 codec is employed. This dataset can be accessed through a link mentioned in the HW2 PowerPoint presentation.

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:**

The S2VT methodology employs a sequence-to-sequence framework for generating captions from videos. This technique uses a hierarchical Long Short-Term Memory (LSTM) structure to process information from both visual and textual sources. Visual data is captured from frame sequences via Convolutional Neural Networks (CNNs), which analyze the frames to provide data in both RGB and optical flow formats. This visual data is then input into the stacked LSTM system, which crafts a word sequence that accurately narrates the video's contents. This strategy ensures that the model comprehensively understands the video's spatial and temporal aspects, enabling it to create precise and informative captions.

- **LSTM structure:**

The model for generating captions from videos utilizes two consecutive Long Short-Term Memory (LSTM) layers to effectively translate a series of frames into a narrative sentence. The initial LSTM layer, highlighted in red, is tasked with interpreting the visual elements from the video sequence, while the subsequent LSTM layer, showcased in green, focuses on constructing the sentence based on the textual input and the video's hidden representation. To frame the sentence structure, the model incorporates specific markers, <BOS> to indicate the start of a sentence and <EOS> for its conclusion. Furthermore, the model employs zero-padding for moments when there is an absence of input during a specific timestep. This method ensures a comprehensive integration of both visual and textual data, culminating in richer and more detailed video captions.

- **Attention Layer:**

The decoder's attention mechanism enables it to focus on different segments of the input sequence selectively throughout the decoding phase. This allows the model, at every step of decoding, to assess and determine the importance of different parts of the input, moving beyond the constraints of a static context vector. The model achieves this by assigning weights to each hidden state of the encoder through the attention process, then creating a composite by calculating a weighted average. Utilizing

this newly formed context vector, the decoder proceeds to produce the subsequent word in the sequence, guided by both the preceding word and the context derived from the vector.

- **Schedule sampling:**

Scheduled sampling is a technique developed to mitigate the "exposure bias" issue encountered during the training of models. Exposure bias arises due to the discrepancy between the data used in training (typically the correct or "ground truth" data) and the data used during inference (generated from the model's own forecasts). Scheduled sampling addresses this by randomly choosing between the ground truth data and the model's previous predictions as the input for training, increasingly favoring the model's predictions as training advances. This method enhances the model's predictive accuracy and reliability by adjusting its exposure to its own outputs.

- **Beam Search:**

Beam search is a widely used search strategy in fields such as natural language processing, where it is employed to generate potential output sequences from a stochastic model. The method operates by maintaining a predefined number of paths, known as the beam width, and calculating the likelihood of every potential next step for each path during each phase.

- I used 'nohup' to run programs in the background. I also created a yml file.

Result:

The average result that obtained is 0.722752652

```
bleu_eval.py  sample_output_testset.txt  testing_data  testing_label.json  training_data  training_label.json
[praguri@node2109 MLDS_hw2_1_data]$ less test_func.py
test_func.py: No such file or directory
[praguri@node2109 MLDS_hw2_1_data]$ cd ..
[praguri@node2109 code_files]$ nano test_func.py
[praguri@node2109 code_files]$ nano hw2_seq2seq.sh
[praguri@node2109 code_files]$ ./hw2_seq2seq.sh
Traceback (most recent call last):
  File "/home/praguri/Deep-Learning/Homework 2/code_files/test_func.py", line 25, in <module>
    test = json.load(open('/MLDS_hw2_1_data/testing_label.json'))
FileNotFoundError: [Errno 2] No such file or directory: '/MLDS_hw2_1_data/testing_label.json'
[praguri@node2109 code_files]$ cd MLDS_hw2_1_data/
[praguri@node2109 MLDS_hw2_1_data]$ ls
bleu_eval.py  sample_output_testset.txt  testing_data  testing_label.json  training_data  training_label.json
[praguri@node2109 MLDS_hw2_1_data]$ cd ..
[praguri@node2109 code_files]$ nano test_func.py
[praguri@node2109 code_files]$ ./hw2_seq2seq.sh
Average bleu score is 0.7227526527805328
```