CPSC – 8430: Deep Learning

ASSIGNMENT – 2

Video Caption Generation

Vinay Kumar Reddy Vallapu Reddy

github

Introduction:

The automatic generation of video captions using natural language has been a difficult task for both natural language processing and computer vision. Because they can model sequence dynamics, Recurrent Neural Networks (RNNs) have been found to be effective in interpreting visual sequences. However, video descriptions must deal with varying lengths of input frames and word sequences.

In contrast to previous studies that used template-based approaches, this model will use the Subject-Object-Verb format and generate text based on the likelihood of correctly matching words. However, due to the limited format, irrelevant text to the corresponding image may appear, and real-time promotion is not possible. To overcome these constraints, a dynamic sequence-to-sequence model will be developed that makes use of Long Short-Term Memory (LSTM). This model will accept frames as input and generate variable-length text while also learning from new images.

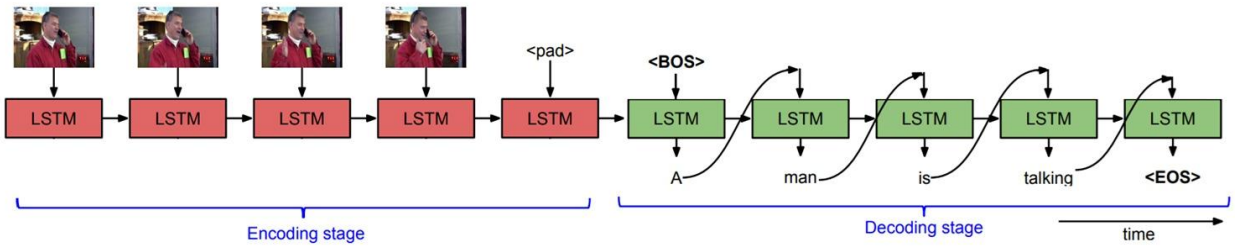
Recurrent neural networks (RNNs) and convolutional neural networks, two recent developments in deep learning techniques, have significantly advanced the area of natural language processing (NLP) (CNNs). Sequence-to-sequence mapping applications like speech recognition and machine translation have demonstrated the great effectiveness of Long Short-Term Memory (LSTM) networks. A stacked LSTM, which reads the input frames one by one, is frequently used to encode the input sequence. The failure to collect any temporal information or the order of the video frames is one of the key drawbacks of employing mean-pooling to create a single feature vector from the CNN output features. Several researchers have proposed a solution to tackle the limitation of temporal dependency using a two-step approach. This approach involves utilizing Conditional Random Fields (CRFs) to generate semantic tuples consisting of activity, object, tool, and location. Then, a Long Short-Term Memory (LSTM) model is used to translate the tuple into a phrase. Another method that can be used to address the issue of temporal dependency is to use a hierarchical recurrent neural encoder that incorporates an additional temporal filter layer comprising LSTM cells . For neural networks to perform better in a range of NLP tasks, such as machine translation, picture captioning, and video captioning, attention methods have gained popularity.

Data Set Features:

MSVD dataset in an experiment where Mechanical Turk workers were asked to provide short descriptions of video clips lasting between 10 to 25 seconds. The dataset consists of approximately 120K sentences, with 1550 video clips in total. The videos were split into two groups for training and testing purposes, with 1450 videos used for training and 100 for testing. Preprocessing was done using pre-trained CNN VGG19, and the features of each video were stored in the format of 80x4096 without reducing any frames from the videos to optimize training data quality.

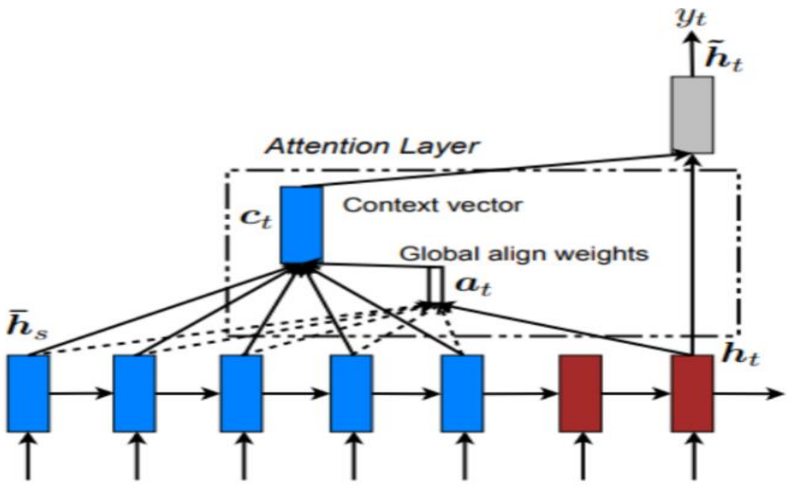
Method Adopted:

The model that is being discussed receives a succession of inputs and generates a sequence of outputs. The pre-trained CNN VGG19 generates an 80x4096 spatial feature map to analyze the video frames. The model's variable sized inputs and variable sized outputs are managed via the LSTM architecture. The decoder is given the ability to choose to focus on the input sequence segments that are most important for creating the output through the implementation of an attention mechanism that facilitates communication between the encoder and decoder components of the model. The model can successfully provide natural language descriptions of video footage because to this combination.



Attention Layer:

by giving the decoder information from each hidden state of the encoder, attention is a process used in machine learning that enables it to focus on pertinent portions of the input sequence. When dealing with lengthy input sequences or when there is a length discrepancy between the input and output sequences, it is very helpful. It has gained popularity in the field of natural language processing and has been used in several projects to enhance the accuracy and fluidity of generated text.



BLEU Score:

The BLEU method is a well-liked tool for assessing the accuracy of material that has been machine translated from one natural language to another. The idea behind it is that a machine translation is better when it is closer to a certified human translation. It offers a quick and easy way to compare the output of machine translations with that of human translations and is one of the most popular, automated, and economical metrics utilized today. Anybody working in the subject of natural language processing, especially those who are creating and assessing machine translation systems, must have this knowledge.

Execution steps:

Preprocessing is an important stage in natural language processing (NLP) that includes converting raw text input into a format suitable for training machine learning models. Common preprocessing techniques include padding, which adds zeros to sequences of varying lengths to ensure they all have the same length, and masking, which ignores certain parts of the input sequence that are not relevant to the model. Preprocessing is a critical step in NLP that involves applying various techniques to transform raw text data into a format that can be used to train machine learning models. Padding and masking are two essential techniques used to ensure that the model can learn from sequences of text data and improve its performance. Other preprocessing techniques include tokenization, stemming, and lemmatization, which help to simplify the data and reduce the vocabulary size. A minimum of three vocabularies is usually recommended for a model to ensure that it can learn from a diverse range of words.

Tokenize:

Dictionary - most frequently word or min count

other tokens：<PAD>, <BOS>, <EOS>, <UNK>

- <PAD> ： Pad the sentencen to the same length
- <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.

I used two object files, 'vid id.obj' and 'dict caption.obj', to hold the dictionary of video id and caption.

For sequence.py execution In Palmetto cluster, I used the below command on my active node:

```
python sequence.py /home/vvallap/HW2/MLDS_hw2_1_data/feat/ /home/vvallap/HW2/MLDS_hw2_1_data/training_label.json
```

```
From 6098 words filtered 2881 words to dictionary with minimum count [3]

Caption dimenstion is: (24232, 2)
Caption's max length is: 40
Average length of the captions: 7.711084516342027
Unique tokens: 6443
ID of 11th video: sRKQfxxEP4M_117_125.avi
Shape of features of 11th video: (80, 4096)
Caption of 11th video: A man is being chewed on by a lion
```

The goal is to train a sequence-to-sequence model with the help of two Python files, sequence.py and train.py. The experiment employs several hyperparameters, which act as parameters to govern the model's learning process. The size and structure of the neural network are determined by the model hyperparameters, while the batch size and learning rate are controlled by the algorithm hyperparameters. The table summarizes the hyperparameters used in the experiment.

HyperParameter	Value
Learning rate	0.001
Use_attention	True
Max_encoder_steps	64
Max_decoder_steps	15
Embedding_size	1024
Batch_size	50
Beam_size	5(if beam search is True)

I used SSH to run a command on the active node of my Palmetto cluster. The command requires two arguments: the first argument is the file path to the feature map in the .npy format, and the second argument is the path to the testing_label.json file that comprises captions for a particular video ID.

```
Epoch# 2, Loss: 2.0751, Average BLEU score: 0.5504, Time taken: 25.87s
Training done for batch:0050/1450
Training done for batch:0100/1450
Training done for batch:0150/1450
Training done for batch:0200/1450
Training done for batch:0250/1450
Training done for batch:0300/1450
Training done for batch:0350/1450
Training done for batch:0400/1450
Training done for batch:0450/1450
Training done for batch:0500/1450
Training done for batch:0550/1450
Training done for batch:0600/1450
Training done for batch:0650/1450
Training done for batch:0700/1450
Training done for batch:0750/1450
Training done for batch:0800/1450
Training done for batch:0850/1450
Training done for batch:0900/1450
Training done for batch:0950/1450
Training done for batch:1000/1450
Training done for batch:1050/1450
Training done for batch:1100/1450
Training done for batch:1150/1450
Training done for batch:1200/1450
Training done for batch:1250/1450
Training done for batch:1300/1450
Training done for batch:1350/1450
Training done for batch:1400/1450
0050/1450
0100/1450
The Average BLEU Score of model is: 0.5504271909423926
Saving model with BLEU Score : 0.5504 ...
Highest [10] BLEU scores:  ['0.6564', '0.6445', '0.5504', '0.5504']
Epoch# 3, Loss: 2.4029, Average BLEU score: 0.5504, Time taken: 25.21s
```

Results:

To calculate Bleu score, I ran the following command

```
python bleu_eval.py vinay_output.txt
```

The average BLEU score was 0.27 before using our model, and it is now 0.55. After 200 epochs, the BLUE score improved to near 0.58.