Anoop Kakkireni
Dr. Feng Luo
CPSC 8430 – Deep Learning 25
March 2022

Homework 2 – Report

Introduction

Generating video captions automatically with natural language has been a difficulty for both the field of natural language processing and computer vision. Recurrent Neural Networks (RNNs), which mimic sequence dynamics, have been shown to be efficient in visual interpretation. A video description must handle a variable length input series of frames as well as a variable length output sequence of words.

The model will be constrained to the format employing Subject, Object, and Verb, as opposed to the template-based approach used in most earlier studies. The result is determined by the probability map of correctly matching the words. Because it restricts itself to a specific order, the text created in this situation may be irrelevant to the image. It can't be encouraged in real time.

The model should be able to learn new photos and be dynamic. So let's create a sequence-to-sequence model that takes frames as input and produces variable-length text output. Long Short Term Memory (LSTM), a form of RNN, will be used to improve the function of the sequence-to-sequence model.

In seq-to-seq applications like speech recognition and machine translation, LSTMs have been shown to be effective. A stacked LSTM does the encoding, which encodes the frames one by one. The LSTM is fed the output of a Convolutional Neural Network applied to each input frame's intensity values. Once all of the frames have been read, the model constructs a phrase word by word.

By pooling the representation of all frames, LSTMs are employed in one of the ways to produce video subtitles. To get a single feature vector, they use mean-pools on CNN output features. The main disadvantage of this method is that it ignores the arrangement of the video frames and fails to change any temporal data.

They adopt a two-step strategy in which CRFs are used to obtain semantic tuples of activity, object, tool, and location, and then an LSTM is used to translate the tuple into a phrase.
A hierarchical recurrent neural encoder is suggested, in which a supplemental temporal filter layer made up of LSTM cells is included during the encoding stage to better denude the temporal dependency of the input frames. To my knowledge, this work has recorded the greatest METEOR score on MSVD data to date, which can be ascribed to the temporal filter layer incorporating higher nonlinearity.

Attention methods have been implemented into several neural networks because they permit salient features to come to the foreground dynamically as needed. Machine translation, picture

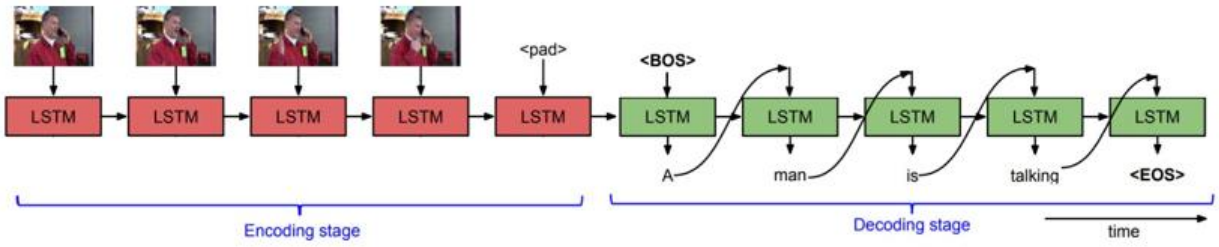
captioning, and video captioning are just a few of the activities that have been observed to benefit from attention models. At the decoder stage of the model, we include an attention layer that can extract useful information from previous word input.

Dataset and features:

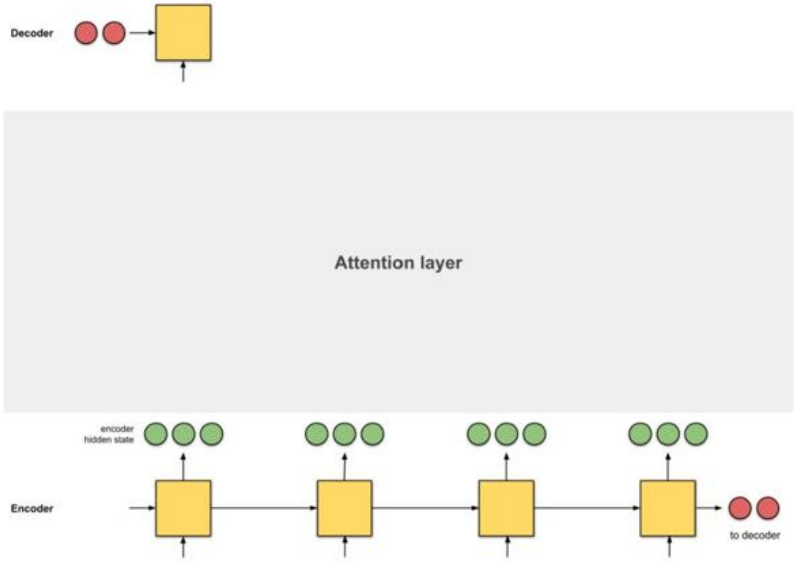
The Microsoft Research Video Description Corpus (MSVD) collection contains around 120K phrases gathered over the summer of 2010. Mechanical Turk workers were paid to watch a brief video clip and then characterize it in a single line. The end result is a set of nearly identical descriptions of over 2,000 video clips. Because the workers were encouraged to perform the job in their preferred language, the data includes both paraphrasing and multilingual alternations.

Method adopted:

- My model is a sequence-to-sequence model that accepts a sequence of video frames as input and outputs a sequence of words.
- The video frames are sent into the pre-trained CNN VGG19, which generates an 804096 spatial feature map for all of the videos. I have created feature maps for each movie in the dataset.
- I used the LSTM Recurrent Neural Network architecture to manage the variable sized input and variable sized outputs. The complete input sequence (x1, x2,..., xn) is first encoded, resulting in a single hidden state vector that may be used to decode a natural language description of the characteristics displayed.



- Attention is a decoder-encoder interface that gives information to the decoder.



BLEU Score:

As per Wikipedia, **Bilingual Evaluation Understudy** is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. Quality is considered to be the correspondence between a machine's output and that of a human: "the closer a machine translation is to a professional human translation, the better it is" – this is the central idea behind BLEU. BLEU was one of the first metrics to claim a high correlation with human judgements of quality and remains one of the most popular automated and inexpensive metrics.

Packages and Libraries:

The following versions of all the technologies were used:

- tensorflow-gpu==1.15.0
- cuda==9.0
- python 3.6.0
- numpy== 1.14
- pandas==0.20

Execution:

Step 1:
In the preprocessing step, padded sequences and maskings are applied to the data. With packed padded sequences, we can use the RNN to solely process the non-padded components of the input sentence. Masking is a technique for making the model ignore some parts we don't want it to see, such as padding elements. This process is carried out in order to improve performance. The training and testing data were obtained from the power point presentation and saved to the 'MLDS hw2 1 data' folder on the local machine. A minimum of three vocabularies should be used.

Step 2: Tokenize

- <pad>: Pad the sentence 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 made use of two object files, 'vid_id.obj',' dict_caption.obj',' to hold the dictionary of id and caption of respective videos.

For Execution of sequence.py I used the following command in Palmetto cluster at my active node:

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

```
(tf_gpu) [akakkir@node0022 VCap]$ python sequence.py /home/akakkir/HW2/MLDS_hw2_1_data/training_data/feat/ /home/akakkir/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
```

Step 3:

The next task is to train the model that was created. We have two python files here. We will build the sequence-to-sequence model with sequence.py. we will train the model with the help of train.py. Please refer to the below table for HyperParameters used in the experiment. A hyperparameter is a parameter whose value is used to control the learning process. An example of a model hyperparameter is the topology and size of a neural network. Examples of algorithm hyperparameters are learning rate and batch size.

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 executed the following command in palmetto cluster on my active node using SSH. The first arg is the path of the feature map which are in the format of .npy file and second arg is the path of the testing_label.json file which has captions for a particular video id.

```
python train.py /home/akakkir/HW2/MLDS_hw2_1_data/testing_data/feat/ /home/akakkir/HW2/MLDS_hw2_1_data/testing_label.json ./anoop_output.txt
```

The calculated Bleu scores after every epoch are shown below.

```
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:

The BLEU score was calculated using the command `python bleu_eval.py anoop_output.txt` and it's corresponding output is shown.

```
(tf_gpu) [akakkir@node0022 VCap]$ python bleu_eval.py anoop_output.txt
Originally, average bleu score is 0.25772300638276563
By another method, average bleu score is 0.5504271909423926
(tf_gpu) [akakkir@node0022 VCap]$
```

The average BLEU score can be seen to be 0.25 originally and using our model it is improved to 0.55. It was observed that after multiple epochs, the BLUE score further improved reaching close to 0.57.