CLEMSON UNIVERSITY

S2101-CPSC-8430 Deep Learning - 001 - 20696

HOME WORK-2

Video caption generation

Submitted By:

Manish Enishetty

To:

Dr. Feng Luo

Abstract

Generating video captions automatically with natural language has been a challenge for both the field of nature language processing and computer vision. The Real time videos will have complex dynamics and methods we use should be sensitive to temporal structure and it should allow input and output of variable length. We will be using Encoder-Decoder framework and MSVD dataset is used for model assessment. We use two Long Short Term Memory (LSTMs) units, one for encoding and other for decoding. An encoder equipped with deep neural networks is used to learn video representation. The decoder on the other hand will generate sentence for the learned representation. For generating efficient captions, we use Beam search algorithm.

1.INTRODUCTION

Generating video captions automatically with natural language has been a challenge for both the field of nature language processing and computer vision [Sequence to Sequence Model for Video Captioning]. In visual interpretation, Re-current Neural Networks (RNNs), which models sequence dynamics, has been proved to be efficient. Image description will try to handle a variable length output sequence of words, while a video description has to handle a variable length input sequence of frames as well as variable length output sequence of words.

Most of the previous works involve template-based approach, the model will be restricted to the particular format using Subject, Object, Verb. The output depends on the probability map of matching the words in a meaning full manner. In this case sometimes the text generated may be irrelevant to the image as it is restricting itself to particular order. In real time it can't be encouraged.

Our model should be dynamic and able to learn new images. So let us build a sequence-to-sequence model that will take the input as frames and generate variable length text output. For better function of sequence-to-sequence model we will be using Long Short Term Memory (LSTM), which is a type of RNN.

LSTM's have been proved to be efficient in seq-to-seq tasks like Speech recognition and Machine translation. Encoding is done by a stacked LSTM which encodes the frames one by one. The output of a Convolutional Neural Network applied to each input frame's intensity values is used as input for the LSTM. The model generates a sentence word by word, once all the frames are read.

In one of the approaches, LSTMs are used to generate video captions by pooling the representation of all frames. They use mean-pools on CNN output features to get a single feature vector. The major drawback of this approach is that it completely ignores the organization of the video frames and fails to manipulate any temporal information.

In other approach, They employ a version of the two-step approach that uses CRFs to obtain semantic tuples of activity, object, tool, and location and then use an LSTM to translate this tuple into a sentence. The model in [8] is applied to the limited domain of cooking videos.

A hierarchical recurrent neural encoder is proposed in where in the encoding stage, a supplemental temporal filter layer composing of LSTM cells is introduced to better denude the temporal dependency of the input frames. To our erudition, this work has reported the highest METEOR score on the MSVD data so far, which can be attributed to the more nonlinearity integrated by the temporal filter layer.

Attention mechanisms have been incorporated into a variety of neural networks, since they sanction salient features to dynamically come to the forefront as needed. Several attention models have been reported to be subsidiary in sundry tasks including machine translation, image captioning and video captioning.

In our model, we implement an attention layer at the decoder stage which can draw utilizable information from anterior word input.

2.Dataset and Features:

The dataset we used in this experiment is MSVD (Microsoft Video Description) dataset. The Microsoft Research Video Description Corpus (MSVD) dataset consists of about 120K sentences. Workers on Mechanical Turk were paid to optically canvass a short video snippet and then summarize the action in a single sentence. The Dataset contains 1550 video snippets each ranging from 10 to 25 seconds. We have taken the split to 1450 and 100 videos for training and testing respectively. We have stored the features of each video in the format of 80×4096 after pre-processing using pre-trained CNN VGG19. During, training we don't reduce any number of frames from video to take full advantage.

1.A person is adding a can of water to a pot that is simmering on a stove.
2.Someone is adding water to a can.
3.A man is adding water to a can.
4.The man put water into the tomato sauce can.

Figure 1: Frames from training video

The above figure is a collection of frames from a sample video in training set.

3.Approach:

Our Model is a seq-to-seq model which takes sequence of video frames as input and generate sequence of words as output.

3.1 Visual Feature Extraction

The Video frames are fed to the pre-trained CNN VGG19 which generates the spatial feature map of all the videos in the format of 80×4096. We have the feature maps generated for every video in the dataset.

3.2 Long Short-Term Memory Networks for Encoding and Decoding

We require to handle the variable sized input and variable sized outputs, we will be utilizing LSTM Recurrent Neural Network architecture. The entire input sequence (x_1, x_2, ..., x_n) is first encoded, summarizing the video into one hidden state vector which is utilized to decode a natural language description of the features shown.

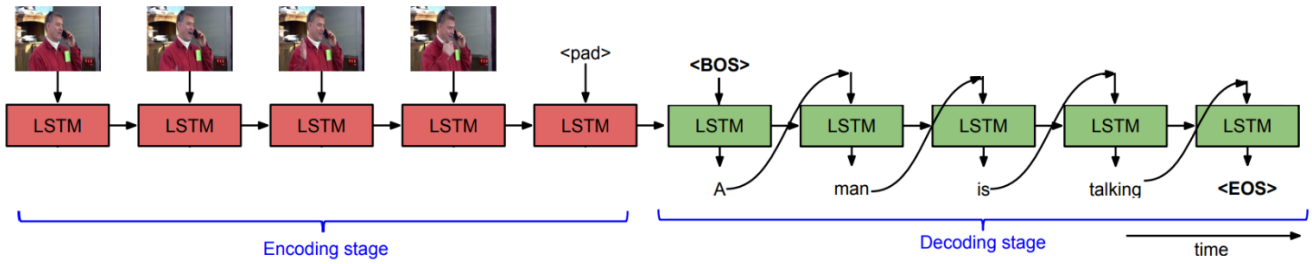

Figure 3.1 Architecture of LSTMs

Attention is an interface between the encoder and decoder that provides the decoder with information from every encoder hidden state.

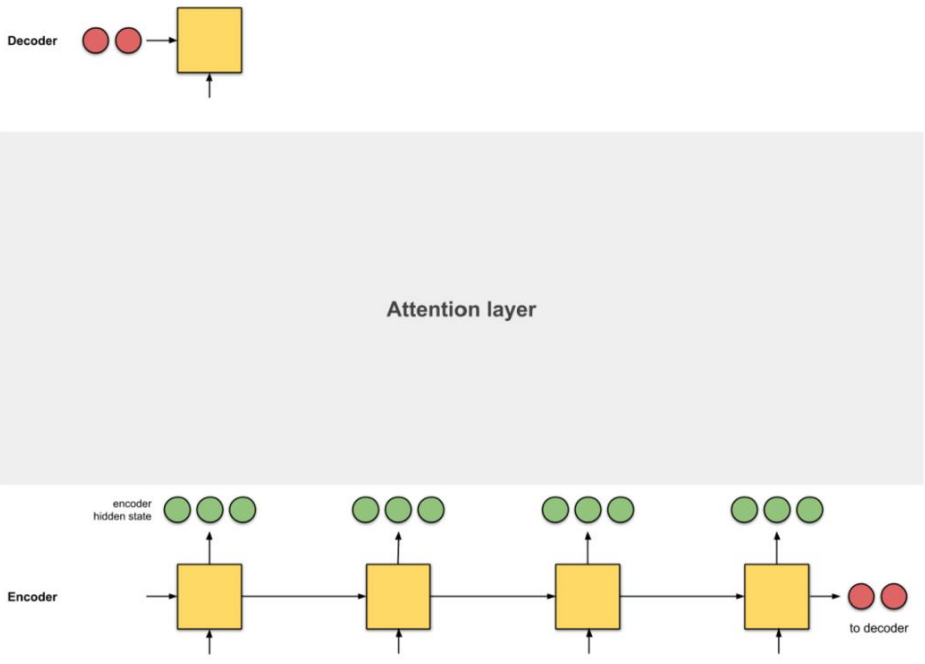

Figure: 3.2 Attention Layer

3.3 BLEU Score

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.
Source: Wikipedia

4. Challenges

Sequence to sequence modeling scope is broader than just the machine translation task. However, there is very little source documents available on the same topic.

There are multiple challenges we can face during implementation:

a. Different attributes of video like action and objects

b. Variable length of I/O

5. All Version Requirement:

We are using the following technologies for functioning of the Model.

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

6. Execution Steps

I. Padded sequences and maskings are added in the preprocess step to the data. Packed padded sequences allow us to only process the non-padded elements of our input sentence with our RNN. Masking is used to force the model to ignore certain elements we do not want it to look at, such as attention over padded elements. This step is executed for performance boosting. The training and testing data were downloaded from the power point

presentation provided and kept in local under ‘MLDS_hw2_1_data’ folder. vocab size should minimum be 3.

II. **Tokenize:**

1. <pad>: Pad the sentence to the same length
2. <bos>: Begin of sentence, a sign to generate the output sentence.
3. <eos>: End of sentence, a sign of the end of the output sentence.
4. <unk>: Use this token when the word isn’t in the dictionary or just ignore the unknown word.

Two object files are build to hold the dictionary of id and caption of respective videos. Object files ‘vid_id.obj’,’ dict_caption.obj’,’ dict_feat.obj’ are being written to build dictionary.

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

python sequence.py /home/menishe/Manish/MLDS_hw2_data/training_data/feat/ /home/menishe/Manish/MLDS_hw2_data/training_label.json

```
(tf_gpu_env) [menishe@login001 Manish]$ python sequence.py /home/menishe/Manish/MLDS_hw2_data/training_data/feat/ /home/menishe/Manish/MLDS_hw2_data/training_label.json
From 6098 words filtered 2881 words to dictionary with minimum count [3]

/home/menishe/.conda/envs/tf_gpu_env/lib/python3.7/site-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a li
r-tuples-or ndarrays with different lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object' when creating the ndarray
  return array(a, dtype, copy=False, order=order)
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: iTA0rWPE4nY_17_23.avi
Shape of features of 11th video: (80, 4096)
Caption of 11th video: A man places chicken into a container
(tf_gpu_env) [menishe@login001 Manish]$
```

ii) Our next task is to train the model we 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 have run the following command in palmetto cluster on my active node using ssh

python train.py /home/menishe/Manish/MLDS_hw2_data/testing_data/feat/ /home/menishe/Manish/MLDS_hw2_data/testing_label.json ./output_testset_manish.txt

The first arg is the path of the feature map which are in the format of .npy file and second arg I sth epath of the testing_label.json file which has captions for a particular video id.

```
Highest [10] BLEU scores:  ['0.5488', '0.5000', '0.4935', '0.4886', '0.4546']
Epoch# 4, Loss: 0.8858, Average BLEU score: 0.5000, Time taken: 28.79s
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
```

I have calculated Bleu score after every epoch. You can see the array of bleu scores in the descending order of the scores.

7.Results

```
(tf_gpu_env) [menishe@login001 Manish]$ python bleu_eval.py test_manish.txt
Originally, average bleu score is 0.2689437917016406
By another method, average bleu score is 0.5760704024416632
(tf_gpu_env) [menishe@login001 Manish]$
```

I have calculated the Bleu score using the following command.

python bleu_eval.py test_manish.txt

you can see the average score is reaching almost 0.57 i.e ~0.6 after 4 epochs

After training the model for 200 Epochs, the bleu_score is nearly to 0.610

8.References

1) http://www.cs.utexas.edu/users/ml/papers/venugopalan.iccv15.pdf
2)https://gist.github.com/vsubhashini/38d087e140854fee4b14

References