Saroj Paudel

CPSC-8430- Deep Learning HWK2 report 3/1/2024

Video Caption Generation

The objective of this assignment is to implement an encoder and decoder model using RNN for video caption generation. The code and relevant files can be accessed at the following repository. The above folder *[HW2/HW2_1]* contains the python file to implement Seq2Seq encoder decoder model along with the file for the trained model for two cases, Greedy search and Beam Search (width, N=2).

https://github.com/espee/CPSC-8430_Saroj_Paudel/tree/main/HW2/HW2_1

The files are as follows:

1. hw2_seq2seq.sh: script file to implement the following command.

   ```
   python model_seq2seq.py MLDS_hw2_1_data output.txt
   ```

2. model_seq2seq.py: Main python file implementing the encoder-decoder model with attention.
3. seq2seqmodel50_Greedy: trained model (50 epochs) for greedy search
4. seq2seqmodel50_Beamsearch2: trained model (50 epochs) implementing beam search with beam width of 2.
5. output_Greedy.txt: Text output for the test data for the greedy model
6. output_BeamSearch2.txt: Text output for the test data for the beam search model
7. vocabulary.pickle: vocabulary data generated from the caption data (Frequency of occurrence > 2)

The major components of the model are briefly described below.

Generating a vocabulary

First, from the given dataset, (input, caption) pairs are computed where input represents the input sequence and caption represents the sequence of words for a caption corresponding to input. In essence, caption is a one-hot encoded sequence over the vocabulary words.

Each input sequence is a sequence of feature vector of size 4096 and corresponds to 80 frames in the input video. The vocabulary is constructed from each word that appears in the caption sequence. A filter for minimum frequency of occurrence is set where the threshold value is set to three resulting in the vocabulary size of 2879. In addition, the vocabulary includes the special tokens <BOS> for beginning of sentence, <EOS> for end of sentence, <PAD> for padding to obtain uniform sized sequence, and <UNK> for representing the words that are not in the vocabulary.

Saroj Paudel
CPSC-8430- Deep Learning HWK2 report 3/1/2024

Seq2Seq Model

The model consists of and Encoder-decoder architecture. Both the sides are implemented using GRUs. Because of computational efficiency, GRU was preferred to LSTM. Both the encoder and decoder have a hidden size of 640. A dropout of 25% is implemented as a means regularization for the encoder, decoder as well as the attention.

Attention module is implemented to identify caption words with greater significance. The attention module is constructed with 1280 hidden units and one output. The first stage of the attention mechanism generates the global attention weights, and the second stage generated the context vector. Similar to the encoder and decoder, a dropout of 25% is implemented for the regularization.

For the decoder, for the first output step, a <BOS> token to signify Beginning of Sentence is used as the initial word. The subsequent GRU decoder units are fed with the immediately preceding decoder output. For example, the second GRU decoder unit is fed with <BOS> as one of its inputs. The number of words for any given caption is set to 20 words. For longer captions it is truncated and for shorter captions padding is done. To minimize the training loss, a supervised method of regularizing, schedule sampling is implemented. Once in a while, at a random timestep, approximately 10% of the time, instead of feeding the predicted output from the preceding GRU unit, the actual output word (from the caption data) is fed to the GRU unit. This is supposed to regularize the model and improve its performance.

Caption generation at the decoder side employ two methods, a greedy search, where the output word is picked based on the highest softmax probability and a beam search method with beam width N = 2, where the two caption candidates are retained and updated at each time step until the final step where the one with higher probability is selected as the output.

Model training and evaluation

The model was trained for 50 epochs. Adam optimizer with learning rate of 1e-3 was used. Figure 1 shows a comparison of the loss for the seq2seq model implementing the greedy method and beam search method (width=2). Although the model with greedy search method seems to perform better than the model with beam search method with fewer training, however, when the epoch was increased, the model with beam search method slightly outperforms the one with greedy method. Although the exact reason for this cannot be verified, the probable reason for this is that the beam search method keeps track of the top two candidates over the decoding time steps and only selects the better of the two candidates at the final step.

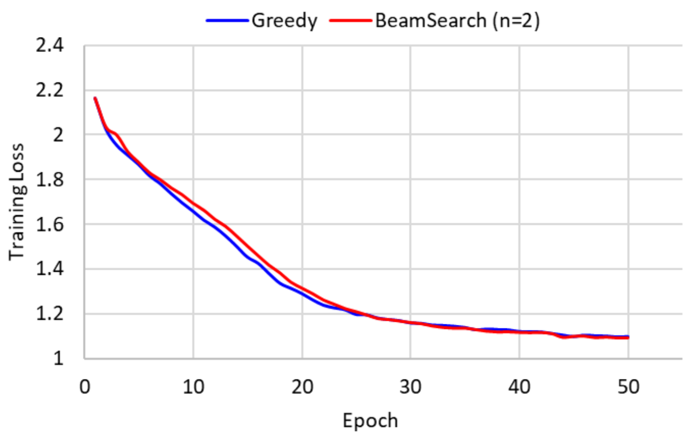
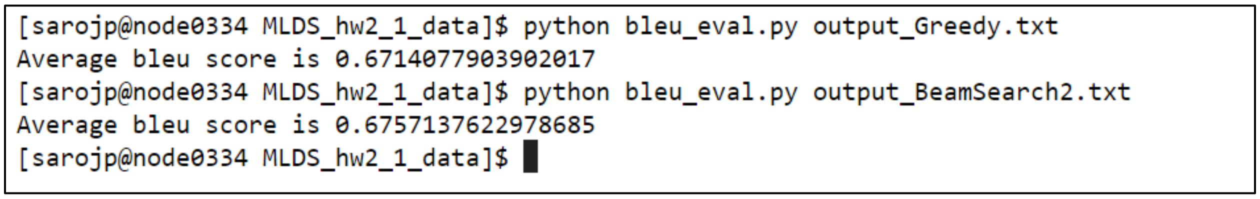


Figure 1. Training loss for the seq2seq model: Greedy search vs beam search method

The beam search method performs better with the testing data as well. This is illustrated by a higher BLEU@1 score compared to the greedy method for the output generated. The BLEU score is shown in the snippet in Figure 2.



```
[sarojp@node0334 MLDS_hw2_1_data]$ python bleu_eval.py output_Greedy.txt
Average bleu score is 0.6714077903902017
[sarojp@node0334 MLDS_hw2_1_data]$ python bleu_eval.py output_BeamSearch2.txt
Average bleu score is 0.6757137622978685
[sarojp@node0334 MLDS_hw2_1_data]$
```

Figure 2. BLEU@1 score comparison for the output of the model implementing the greedy method (upper) and the beam search method (width =2)