CPSC 8430 DEEP LEARNING HOMEWORK 2 REPORT
VIDEO CAPTION GENERATION

SATHWIKA ADIMULAM
Dr. Feng Luo

Github link :- https://github.com/sathwika-adimulam/8430_DeepLearning_HW2

Problem Statement

The goal is to generate natural language captions for a video using a sequential-to-sequential approach. The input to the program is a video, and the output is a sequence of captions that describe the actions in the video.

Dataset

The MSVD dataset is a popular dataset that is commonly used for training and testing video captioning models. The dataset consists of 1550 short video clips, each lasting between 10 and 25 seconds. Of these clips, 1450 are used for training the models, and the remaining 100 are used for testing. The dataset also includes labels for both training and testing. During training, the models are trained using .json files that contain the labels for the corresponding video clips. These labels are used to teach the models to accurately describe the actions and events in the videos.

Abstract

Generating natural language captions for video content is to use a sequential-to-sequential model that consists of an encoder and decoder architecture. The MSVD dataset, which comprises short videos and their associated textual descriptions, is commonly used to evaluate video captioning models. During training, the model learns to predict the next word in a sequence of words that describes the video, based on the visual features extracted from the video frames and the previous words in the sequence. To improve performance, attention mechanisms are incorporated to enable the model to focus on relevant parts of the video at each time step. Additionally, schedule sampling is used during training to mitigate the issue of exposure bias, where the model's performance during inference is worse than during training due to being trained on ground-truth captions. Overall, the sequential-to-sequential model, attention mechanisms, and schedule sampling technique provide an effective framework for generating natural language captions for video content.

Requirements

1.Python
2.Torch
3.Scipy
4.Numpy

5.Pandas
6.MSVD dataset

Dictionaries

In the context of seq2seq models, a dictionary is an essential element. It allows for the mapping of words to indices and vice versa. This functionality is enabled by assigning a unique index to each word in the vocabulary. A few of the tokens used to hold data in the dictionary are listed below:

- Dictionary: most used words or minimum count
- other tokens PAD, BOS, EOS, UNK
    - <PAD> : Lengthen the sentence by the same amount.
    - <BOS> : Beginning of sentence is a signal to create the output sentence.
    - <EOS> : End of sentence, a signal that the output sentence has ended.
    - <UNK> : Ignore the word if it isn't in the dictionary or use this token when it isn't.

Model Execution:

Base model

The video captioning model is built using two layers of GRU or Gated Recurrent Units, which are simpler and faster than LSTM or Long Short-Term Memory units. Since the dataset contains relatively short video sequences, the GRU layers can deliver accurate results with less computational overhead. The sequence-to-sequence or S2VT model consists of an encoder RNN and a decoder RNN, both implemented using GRU layers. The encoderRNN class processes the input video frames and encodes them into a condensed form, which is then passed to the decoderRNN class. The decoderRNN class generates the output captions by decoding the encoded information and separating the captions using sentence tokens. During decoding, the model utilizes the encoderRNN to extract features from the video and the decoderRNN to generate captions based on the extracted features. Overall, the two-layer GRU-based architecture and the encoder-decoder RNN framework enable the model to accurately generate captions for input videos.

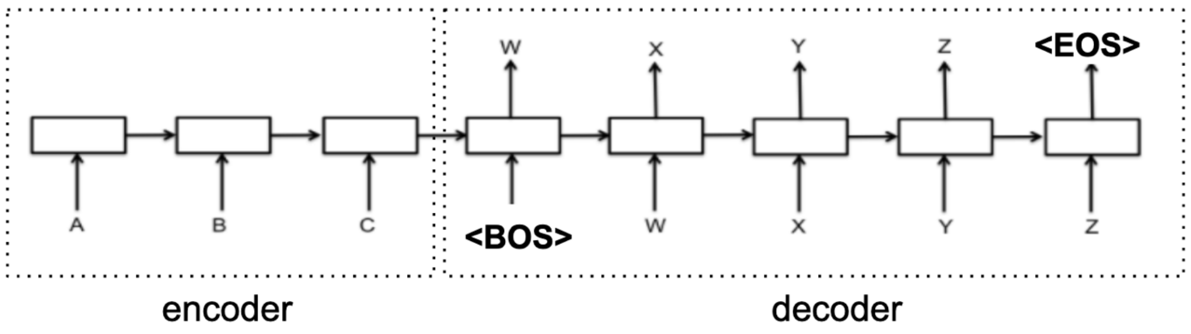

Figure 1

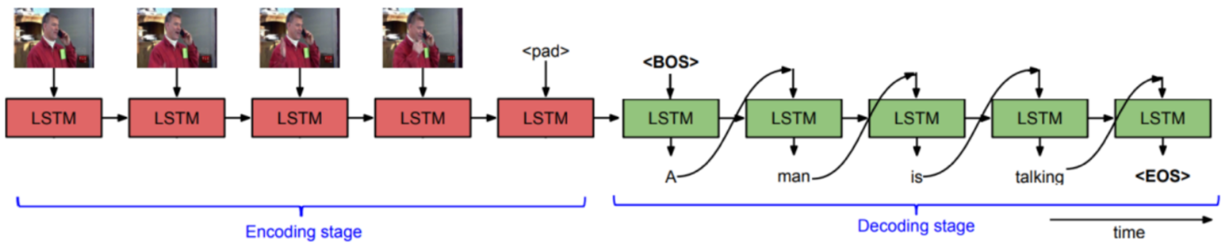

Figure 2

Attention layer

In a sequence-to-sequence model for natural language processing, the attention layer helps the model to focus on different parts of the input sequence while decoding. To do so, the attention layer takes two inputs: the hidden state of the decoder and the output of the encoder, which captures the input sequence in a condensed form. The attention layer then calculates a score for each encoder output vector based on its similarity to the decoder hidden state. The scores are transformed into attention weights using a softmax function, which determines how much attention the model should pay to each encoder output vector. The resulting weighted sum of the encoder output vectors is used to update the decoder hidden state and generate the output for the current decoding time step. By attending selectively to different parts of the input sequence at each decoding time step, the attention layer enables the model to generate more accurate and relevant outputs.

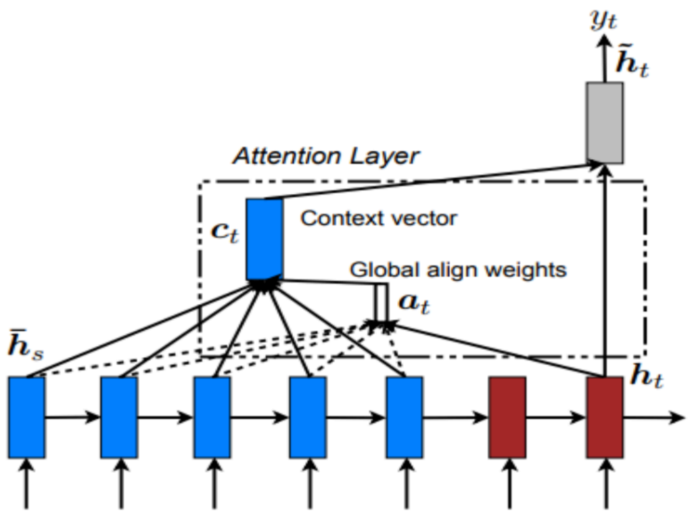

Figure 3

Schedule sampling

During training of video caption generation models, the captions generated by the model are compared to the ground truth captions to calculate the loss and update the model parameters. However, this can lead to exposure bias during inference, where the model's performance may be significantly worse than during training, as it has not learned to handle the discrepancies between the predicted and actual captions. To address this issue, schedule sampling is used during training. Schedule sampling is a technique where the model is fed either its own predictions or the ground truth captions at each time step during training. This enables the model to learn to generate accurate captions even when it deviates from the ground truth, resulting in better performance during inference.

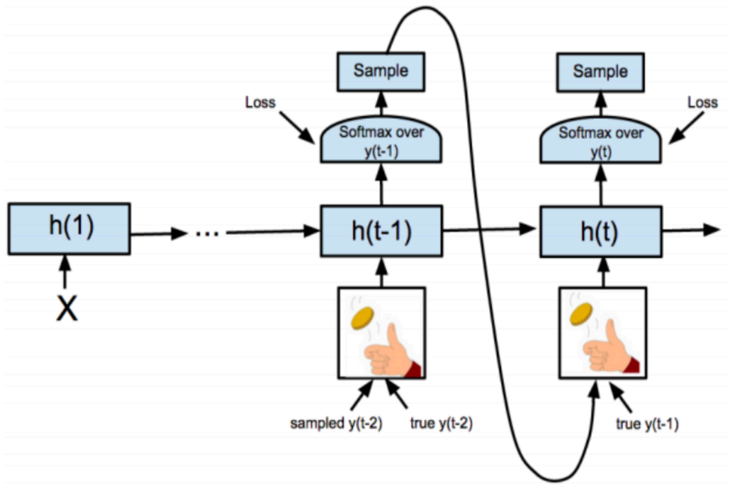

Figure 4

Model Parameters

- Epochs = 80, 100
- Train dataset size = 1450
- Test dataset size = 100
- Learning Rate = 0.001
- Batch Size = 128
- Hidden layers size = 512
- Optimizer = Adam Optimizer
- Dropout = 0.3
- Teacher Learning Ratio = 0.7
- VocabSize = n>3

Evaluation

The Average bleu score is 0.680745 for testing data using 80 epochs
The Average bleu score is 0.691942 for testing data using 100 epochs