Deep Learning - CPSC 8430

HW-2

Name: Venkata Subbu Sai Hiranmayee Machavolu

GitHub link of my code: https://github.com/hiranmayee1123/Deep_Learning/tree/main/HW2_1

Introduction

Video caption generation is the task of automatically producing a textual description for a given video. This report focuses on developing a sequence-to-sequence (seq2seq) model with an attention mechanism to generate captions for input videos. The seq2seq architecture, commonly used in natural language processing, processes sequential data by encoding video frames and generating corresponding text. The attention mechanism enhances the model's ability to focus on different sections of the video at each time step, leading to more accurate and contextually relevant captions. This task presents challenges such as handling diverse video attributes, varying sequence lengths, and ensuring meaningful output, which are addressed through the use of recurrent neural networks (RNNs), attention, and advanced training techniques.

1. Approach

To generate captions for video input, the approach consists of several key components:

- **Data Pre-processor**
- **Base Model (S2VT)**
- **Encoder and Decoder Layers**
- **Attention Layer**
- **Training and Testing Procedures**
- **BLEU Score Evaluation**

These components are organized across the following scripts:

- **Main Script**:
    - Implements the data pre-processing, base model (S2VT), and training process.
- **Models Script**:
    - Contains the Encoder and Decoder layers, along with the Attention layer.
- **Testing Script**:
    - Responsible for testing the model and calculating the BLEU score.
- **Hw2_seq2seq.sh Script**:

- A shell script that executes the testing process, taking the test data directory and test output file location as inputs.

2. Data Pre-processor

The data pre-processor is responsible for reading all the training video features and corresponding captions, storing them in a feature dictionary and a caption dictionary.

- **Dictionary**:
  - This function reads the caption data from a JSON file and identifies key words. The dictionary stores all the English captions associated with the videos, where a single video may have multiple captions. To build the dictionary, the captions are analyzed, and words that exceed a certain frequency threshold (appearing more than three times) are included. Words that occur less frequently are replaced with an unknown token ("<UNK>") to minimize the dictionary size and focus on essential terms.
  - This process is crucial since raw sentences cannot be fed directly into the model. The captions are tokenized into individual words, which are then converted into their respective numerical indices. This transformation allows the model to effectively process and learn from the input data.
- **Tokens used in the dictionary:**

  <PAD>: This token is used to pad sentences to ensure that all input sequences are of equal length, making them uniform for processing by the model. Padding helps to avoid issues when working with sentences of varying lengths by filling shorter sentences with the <PAD> token, so all sentences are treated with the same number of time steps during training and inference.

  <BOS>: The "Beginning of Sentence" token is added at the start of each caption to indicate where the output sequence generation should begin. This token provides a clear signal to the decoder model, ensuring it knows when to start generating the corresponding caption for a video.

  <EOS>: The "End of Sentence" token is used to signal the end of an output sequence. The decoder model uses this token to know when the caption generation is complete, preventing unnecessary or incorrect word generation beyond the intended caption length.

  <UNK>: The "Unknown" token represents words that do not appear frequently enough to meet the threshold for inclusion in the dictionary. If a word appears less than the required number of times in the captions, it is replaced with the <UNK> token, which simplifies the vocabulary by focusing on more important or frequently used words. This ensures that the model is not overloaded with rare or less meaningful words.

- **Data Processor:**
  - The data processor is a crucial function that manages the interaction between the video features and their corresponding captions. It begins by loading video labels (identifiers for the videos) and extracting the corresponding feature sets based on these video IDs. The video features are pre-extracted and provided, and each video is associated with a set of labels that describe its content (such as objects, actions, etc.).
  - Once the video features and labels are loaded, the processor feeds the captions into the previously generated dictionary. The captions are converted from sentences to numerical representations based on the tokens created earlier (such as <BOS>, <EOS>, <PAD>, <UNK>, and regular words). These numerical representations are then transformed into tensor formats, which are multidimensional arrays suitable for machine learning models like RNNs or LSTMs.
  - The tensor outputs for both video features and captions are prepared in this step, ensuring that the model can efficiently process them during training and testing phases. This conversion of both the video and text data into a machine-readable format is essential for the sequence-to-sequence learning process, enabling the model to learn the relationships between video sequences and their corresponding textual descriptions.

3. Models:

- **Encoder and Decoder:**
  - This script implements a **sequence-to-sequence (seq2seq)** model using **2-layered GRU** (Gated Recurrent Units) networks for both the encoder and decoder. The encoder processes and encodes the video features into a latent representation in its first layer (referred to as **encoderRNN**). This encoded representation is then passed to the decoder (referred to as **decoderRNN**), which generates the corresponding output, in this case, captions.
  - The decoding process begins with special tokens such as **<BOS>** (Beginning of Sentence) and **<EOS>** (End of Sentence), which mark the start and end of the generated captions, respectively. These tokens help the model know when to start and stop generating the caption text. The decoder then processes the video encoding to generate the appropriate English words describing the video content.
  - Dropout with a rate of **0.3** is used to regularize the model and prevent overfitting. This means that during training, 30% of the neurons are randomly "dropped" from the network to ensure the model generalizes well to unseen data.

  The structure of the GRU network is straightforward:

  - Two layers are stacked for both the encoder and decoder.

- GRUs, a variant of RNNs, are chosen because they can efficiently handle sequence data while mitigating the vanishing gradient problem typically encountered in standard RNNs. GRUs achieve this by using gates that control the flow of information through the network, allowing it to retain or discard information as necessary.

- **Attention Layer:**
  - The **Attention Layer** is added to enhance the performance of the seq2seq model. In traditional seq2seq models, the encoder compresses all the input information into a fixed-length vector, which can make it challenging for the decoder to generate accurate outputs for long or complex sequences. Attention addresses this issue by allowing the decoder to "attend" to different parts of the input at each time step of the decoding process.
  - At each decoding step, the model can focus on various regions of the input sequence, which, in this case, are the hidden states from the encoder. This mechanism improves the model's ability to generate more accurate captions by selectively attending to relevant parts of the video when generating each word in the output.

  The attention mechanism works as follows:

  - The decoder's current hidden state is compared to the encoder's output (hidden states), and a **matching score** is computed for each part of the input. This score is essentially a scalar value representing how relevant each part of the encoder's output is to the current decoding step.
  - These scores are then processed through a **softmax function**, which converts them into probabilities. The decoder uses these probabilities to weight the encoder's hidden states, effectively deciding which parts of the input to focus on.
  - The weighted sum of the encoder’s hidden states is then combined with the decoder’s hidden state to produce the final output for the current time step.
  - The decoder's updated hidden state is passed to the next time step, where the process repeats until the entire caption is generated.

This attention mechanism allows the model to dynamically select relevant parts of the video at each word generation step, improving the accuracy and contextual relevance of the generated captions.