CSE 4392 Special Topic: Natural Language Processing

Homework 9 - Spring 2024

Due Date: Apr 9th, 2024, 11:59 p.m. Central Standard Time

This week, we explored Recurrent Neural Networks (RNNs) and their variations, focusing on their implementation in Natural Language Processing (NLP). We covered the fundamentals of RNNs and their limitations like the vanishing gradient problem, and introduced Long Short-Term Memory (LSTM) networks as a solution. Moreover, we discussed Bidirectional RNNs (BiRNNs), which process input sequences in both forward and backward directions to capture context from past and future inputs. Finally, we applied these concepts to understand how RNNs, LSTMs, and BiRNNs can be effectively utilized in various tasks.

Problem 1 - 100%

This week's homework delves into the realm of Recurrent Neural Networks (RNNs) and their variants, focusing on their application in text classification tasks(MBTI Classification). The assignment comprises two questions aimed at reinforcing your understanding of RNNs and Bi-directional RNNs (BiRNNs) for text classification:

Question 1 - 40%

In this question, you'll undertake the mathematical derivation of the gradients for parameters within a Bi-directional RNN model tailored for text classification. Your task involves computing gradients with respect to the model's parameters, including input-to-hidden weights, hidden-to-hidden weights, and biases.

Recall we learned the derivation of the gradients for RNNs as:

$$\frac{\partial L}{\partial \mathbf{W}} = -\frac{1}{n}\sum_{t=1}^{n}\sum_{k=1}^{t}\frac{\partial L_t}{\partial \mathbf{h}_t}\left(\prod_{j=k+1}^{t}\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}\right)\frac{\partial \mathbf{h}_k}{\partial \mathbf{W}},$$

If we follow the implementation of BiRNNs as below:

$$f_1(\mathbf{h}, \mathbf{x}) = g(\mathbf{W}_1\mathbf{h} + \mathbf{U}_1\mathbf{x} + b_1) \tag{1}$$
$$f_2(\mathbf{h}, \mathbf{x}) = g(\mathbf{W}_2\mathbf{h} + \mathbf{U}_2\mathbf{x} + b_2) \tag{2}$$
$$\overrightarrow{\mathbf{h}}_t = f_1(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t), t = 1, 2, ..., n \tag{3}$$
$$\overleftarrow{\mathbf{h}}_t = f_2(\overrightarrow{\mathbf{h}}_{t-1}, \mathbf{x}_t), t = n, n-1, ..., 1 \tag{4}$$
$$\mathbf{h}_t = \left[\overrightarrow{\mathbf{h}_t}, \overleftarrow{\mathbf{h}_t}\right] \in \mathbb{R}^{2d} \tag{5}$$
$$P(y_i = k) = softmax_k(\mathbf{W}_o\mathbf{h}_t) \tag{6}$$
$$L = -\frac{1}{n}\sum_{n}^{i=1} \log P(y_i = k) \tag{7}$$

Please derive the gradients for parameters $\mathbf{W}_0$, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{b}_1$, $\mathbf{b}_2$, in a Bi-directional RNN model.

Question 2 - 60%

Hope your final project is going well! For implementing your final project, you must have a taste of deep learning platforms like PyTorch.

In this question, you are tasked with implementing a bidirectional Long Short-Term Memory (LSTM) model for text classification using PyTorch.

Although we just learned the Bi-directional RNNs in class, you can easily implement a Bi-directional LSTM model in PyTorch. The architecture of the BiLSTM model is shown as below:

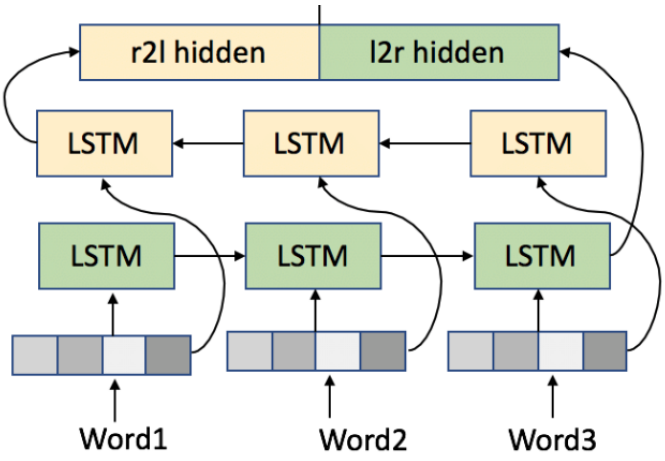

Figure 1: Bidirectional LSTM Model Architecture

However, **you are not allowed to use PyTorch's built-in bidirectional LSTM module**, which means you cannot use the `torch.nn.LSTM` module with the `bidirectional=True` flag. Instead, you will manually implement the bidirectional functionality by concatenating the outputs of two separate LSTM layers: one processing the input sequence in the forward direction and the other in the backward direction.

Description:

1. **Define the Model Architecture:** Implement the `BiLSTM` function, which takes input parameters such as `input_size`, `hidden_size`, `num_layers`, and `output_size`. Inside this function, define two separate LSTM layers (`lstm_forward` and `lstm_backward`) to process the input sequence in forward and backward directions, respectively.

2. **Forward Pass:** Define the forward pass function inside the `BiLSTM` function. Initialize the hidden and cell states for both forward and backward LSTMs, perform the forward pass with the forward LSTM, reverse the input sequence, perform the forward pass with the backward LSTM, and concatenate the outputs of both LSTMs along the feature dimension.

3. **Output Layer:** Add a linear layer (`fc`) to transform the concatenated output of the bidirectional LSTMs to the desired output size (number of classes for classification).

4. **Training Loop:** Instantiate the `BiLSTM` model with appropriate parameters. Define the loss function (e.g., CrossEntropyLoss) and optimizer (e.g., Adam). Train the model on the provided dataset using a suitable training loop, optimizing the model parameters to minimize the loss.

5. **Evaluation:** After training, evaluate the model's performance on a separate validation or test dataset. Calculate relevant evaluation metrics such as accuracy, precision, recall, and F1-score to assess the model's classification performance.

Attach your codes and report.