ASSIGNMENT 4
BERT MODEL FOR SENTIMENT CLASSIFICATION

CS 478 NATURAL LANGUAGE PROCESSING (FALL 2024)
https://nlp.cs.gmu.edu/course/cs478-fall24/

OUT: Oct 16, 2024

DUE: Oct 30, 2024

TOTAL CREDITS: 100 Points

Your name: Arron Birhanu

Your GID: G01315277

Shareable link to your notebook:
https://colab.research.google.com/drive/179xdfls6qDGXt5KWsXSWWqGa-bD_uktp?usp=sharing
(Make sure to submit your notebook to Blackboard as well)

Academic Honesty: Please note that students should complete the assignment independently. While you may discuss the assignment with other students, **all work you submit must be your own!** In addition, if you use any external resources for the assignment, you must include references to these resources at the end of this PDF. However, **you are NOT allowed to use AI assistants such as ChatGPT and Claude to complete your assignment, although using them for generic concept understanding is okay.**

Overview and Submission Guideline: In this project, you will learn to use a pre-trained and then fine-tuned BERT model for binary sentiment classification. The assignment also comes with a Jupyter notebook (https://jupyter.org/). **Since you will fine-tune the BERT model in Part 2, you are suggested to use Google Colab, which provides free GPU resources, to complete this homework.** Completing it locally on your laptop/computer is also feasible, but it requires a longer time for BERT fine-tuning. You will be using exactly the same dataset as in Assignment 3. Like in Assignment 3, complete the notebook, save your edits *(make sure to save the cell outputs as well)*, add a shareable link to your notebook above, and submit your notebook to Blackboard as a backup.
How to fill out this PDF? In most answer blanks, you only need to provide the execution output of your code implementation. There are only a few blanks that ask you to provide your code implementation for ease of grading. When you complete the PDF, submit it to Gradescope.

What and Where to Submit? To summarize, you will submit three items from this assignment:

- A completed PDF compiled from this LaTex source file to Gradescope.
- Your completed notebook to Blackboard.
- Your test outputs to Blackboard.

Part 1: Evaluate a Pre-Trained Sentiment Classifier (70 Points)

In this part, we will evaluate a pre-trained sentiment classifier ("textattack/bert-base-uncased-yelp-polarity") on our sentiment classification dataset. This classifier, as shown by its name, was pre-trained on BERT (uncased base version) and then fine-tuned on a Yelp polarity dataset. It has two class labels, 0 representing negative and 1 representing positive. The model is open-sourced on the transformers model hub (https://huggingface.co/textattack/bert-base-uncased-yelp-polarity). We will test how well the model performs on the dev set, without fine-tuning.

Q1: Follow the example here, create a tokenizer and load the pre-trained model of "textattack/bert-base-uncased-yelp-polarity" (10 points).

Your Code Solution

```
# Load the tokenizer
tokenizer = BertTokenizer.from_pretrained(pretrained_checkpoint)

# Load the pre-trained model
model = BertForSequenceClassification.from_pretrained(
    pretrained_checkpoint)
```

Q2: Tokenize the sentence using the created tokenizer (10 points):

Your Code Solution

```
inputs = tokenizer(dev_exs[0].sentence, padding=True, truncation=True,
     return_tensors="pt")
```

Q3: Now, run the model inference given this sentence. You should use the "inputs" defined above. The returned "outputs" should contain the model logits (10 points):

Your Code Solution

```
# Run the model inference
inputs = tokenizer(dev_exs[0].sentence, padding=True, truncation=True,
     return_tensors="pt").to(device)  # Move inputs to the same device

# Perform inference with the model
with torch.no_grad():  # Disable gradient calculation
    outputs = model(**inputs)  # Get model outputs
```

Q4: Calculate the prediction labels (1 for positive and 0 for negative) based on the model logits (15 points):

Your Code Solution

```
# Get the predicted class ID from the logits
predicted_class_id = torch.argmax(outputs.logits, dim=1).item()  # Get the index of the max logit and convert to Python int
print("Prediction:", predicted_class_id)
```

Provide your code output below:

Your Code Solution

```
Prediction: 1
```

Q5: While the above code can process a single input sentence, to make the best use of the GPU, we want to batchify this process, including tokenizing a batch of input sentences, and running the loaded classifier to make predictions for the batch data. Show your code solution for the "TODO" part below (10 points).

Your Code Solution

```
# TODO: tokenize the `batch_exs` data. The resulting `batch_inputs` will have exactly the same fields as `inputs`, i.e., input_ids, token_type_ids, and attention_mask. However, you should see that now each field contains the result from a batch of data with paddings.
# Hint: https://huggingface.co/course/chapter2/2?fw=pt#preprocessing-with-a-tokenizer
batch_inputs = tokenizer(
                [ex.sentence for ex in batch_exs],  # List of sentences to tokenize
                padding=True,                       # Pad to the maximum length in the batch
                truncation=True,                    # Truncate to the model's maximum length
                return_tensors="pt"                 # Return as PyTorch tensors
            ).to(device)  # Move the inputs to the appropriate device
```

Q6: Show your code solution for the "TODO" part for batch prediction (10 points):

Your Code Solution

```python
def batch_predict(classifier, batch_inputs: torch.Tensor) -> List[int]:
    # Run the classifier on the batch inputs
    with torch.no_grad():  # Disable gradient calculation for inference
        outputs = classifier(**batch_inputs)  # unpack the inputs
        logits = outputs.logits  # Extract the logits

    # Get the predicted class labels (0/1)
    preds = torch.argmax(logits, dim=1).cpu().numpy().tolist()  # Convert to numpy and then to list
    return preds
```

Q7: Now, let's run the model and evaluate its performance on the dev set! (If your implementation is correct, you should see an accuracy of around 0.8.) Show your result below (5 points).

Accuracy	Precision	Recall
0.800459	0.796053	0.817568

Part 2: Fine-tune a BERT-based Model for Sentiment Classification (30 Points)

While in the last part we directly load in a sentiment classifier and use it in our movie review sentiment classification task, in this part, we will fine-tune this classifier on our dataset and see if it can achieve a better task performance in the end.

Q8: Complete the "TODO" for implementing BERT pre-training (15 points).

Your Code Solution

```
# TODO: call the model and compute the loss
# Hint: you don't need to implement the loss on your own when using the transformers 'BertForSequenceClassification';
# see https://huggingface.co/docs/transformers/v4.25.1/en/model_doc/bert#transformers.BertForSequenceClassification.forward.example for help
batch_outputs = model(**batch_inputs, labels=batch_labels)  # Include labels here
loss = batch_outputs.loss  # Get the loss from the output
```

Q9: Describe your observation. (10 points)

Report the best performance of your model and the epoch you obtained it.

Best Acc	Epoch
.89	1

How does the fine-tuned BERT model compare with (a) its counterpart without fine-tuning and (b) the feedforward neural network that we trained in the previous assignment? Why do you think the fine-tuned BERT could have the observed performance?

Your Answers

Typically, the BERT models outperform models that are not as fine-tuned and feedforward neural network in sentiment classification processes. This result is attributed to contextual embeddings, levarage transfer learning, attention mechanisms, and other language patterns found through the deep architecture. These patterns are widely present in the transformers and pytorch library. As such, they produce a combination of factors to give fine-tuned BERT an edge over other engines regarding comprehending and classifying text data.

Q10: The last code block runs your fine-tuned BERT to predict labels for the test examples. Submit your test output (i.e., your predictions) to Blackboard. (5 points)

Post-Assignment Questions (Required)

Q-P1: Did you use any resources (e.g., online tutorials, blog posts, etc.) while completing this assignment? If you did, please list them below:

Your Answer

I used the documentation for pytorch and transformers to properly handle the given libraries. Nothing else.

Q-P2: Did you use AI assistants (e.g., ChatGPT) while completing this assignment? If you did, please describe how it was used. Note that you are NOT allowed to use the AI assistant's code output as yours, but it is okay to use it to assist you in generic concept understanding.

Your Answer

None