Assignment 4

Practical Deep Learning for Language Processing

Dr. Aseem Behl
School of Business and Economics
University of Tübingen

Friday, 17. February 2023

Fine-Tuning GPT-2

Part A - Fine-tuning GPT-2 for English to Latex conversion

In part A, we will improve upon the English to Latex converter we trained in tutorial session for chapter 13. Specifically, before we fine-tune the the GPT-2 model on our training data of 40 supervised training examples, we would first pre-train it on mathematical text by letting the model read a calculus book. This pre-training uses the self-supervision provided by the calculus text book to fine-tune the GPT-2 model on the task of Causal language modeling for both the English natural language and the Latex markup language. Specifically, you will implement the following steps:

1. Download and import the `calculus made easy.txt` from the ILIAS assignment folder.
2. Initialise a `DataCollatorForLanguageModeling`. Set the `mlm` attribute of the `DataCollatorForLanguageModeling` to `False`. The `mlm` attribute controls whether or not to use masked language modeling. There are two types of language modeling, causal and masked. Causal language modeling predicts the next token in a sequence of tokens, and the model can only attend to tokens on the left. This means the model cannot see future tokens. GPT-2 is an example of a causal language model. Masked language modeling predicts a masked token in a sequence, and the model can attend to tokens bidirectionally. This means the model has full access to the tokens on the left and right. BERT is an example of a masked language model.

3. Initialise a pertained `GPT2Tokenizer` for '`gpt2`'.
4. Import the book using the `TextDataset` class from the `transformers` package. `TextDataset` reads the full input text, tokenizes it and cuts it in block_sized chunks. Then adds special tokens. While initialising the `TextDataset`, use the tokeniser initialised in the last step. Use `block_size=32`.
5. Split the book dataset into `80%` training examples, and `20%` evaluation examples.
6. Import a pertained `GPT2LMHeadModel` for `'gpt2'`.
7. Initialise `TrainingArguments` object. Set the batch size for training and evaluation equal to 32. Set the epochs making sure that the GPT2 model reads the books only once.
8. Initialise the `Trainer` object. Train, evaluate and save your model.
9. Next you will replicate the steps from the tutorial, only difference being that we will start with the `GPT2LMHeadModel` we just pre-trained on the calculus textbooks.
10. Download and import the `english_to_latex.csv` from ILIAS assignment page.
11. As shown in the tutorial, create training examples for training the GPT-2 by transforming the training data using a *Conversion Prompt* and *Conversion Token*. The *Conversion Prompt* informs the GPT-2 model about the task we want it perform. The *Conversion Token* indicates to the model to start generating the solution to the task.
12. Convert the training examples to a `Dataset` object. The easiest way to create this is using the `Dataset.from_pandas` function.
13. Split the dataset into `80%` training examples, and `20%` evaluation examples.
14. Tokenise the dataset using the earlier imported tokeniser.

15. Initialise `TrainingArguments` object. Set the batch size for training equal to 2, and number of epochs equal to 10. Write in one line why it might be helpful to use a smaller training batch size here.
16. Initialise the Trainer object. Train, evaluate and save your model.
17. Test your model by generating the Latex for the following two mathematical statements:

    'f of x equals integral from 0 to pi of x to the fourth power'

    'f of x is sum from 0 to x of x squared'

Part B - Teaching GPT multiple tasks at once

In the previous exercise, we introduced a new kind of prompt to GPT-2 for the English to Latex conversion. But what if we took it even one step further, can we teach GPT-2 multiple tasks at once in the same training loop? To do that, we're gonna use a data set called Amazon Fine Food Reviews, which contains 50,000 food reviews from Amazon. We are going to use this dataset for training the model on two tasks: sentiment analysis task and a separate prompt for summarisation task. Specifically, the process will look like this:

1. Download and import the `reviews.csv` from ILIAS assignment page.
2. The CSV here has only three columns, a text column, a summary column, and a score column, score being a score from one to five. Our job is going to be to engineer prompts for each task that we are trying to perform so that each row in the original dataset will end up being two data points in our fine tuning process, one for summarization, one for sentiment analysis.
3. We'll remove very short and very long summaries and leep only reviews whose summary length is between 10 and 25 characters. The short length

of summaries is going to make it easier for our GPT-2 model to learn how to summarize.

4. Also, instead of predicting the score, we would like to predict a sentiment: positive, negative, or neutral. So we are going to map our score label to be a sentiment label where it'll be positive if the score was four or five, neutral, if it was three, and then negative if it were one or two.
5. To balance our dataset, we will take `1,000` examples from each sentiment group, and that will be our final data set.
6. We'll be using a distilled version of GPT-2, `distilgpt2`. It's a smaller model, so it'll be faster to train, faster to predict, and it should be almost as performant as our baseline GPT-2 model.
7. So again, we're going to load up our tokenizer for `distilgpt2`.
8. Next, we're going to create the training texts for the two tasks. This is very similar to our latex exercise, but instead of one prompt, we're going to be inputing GPT-2 with two prompts during training. For the sentiment prediction task the training text is going be the following: prepend our sentiment prompt, put in our review, our review indicator, and then at the end, we'll have our sentiment token to tell it to start creating a sentiment prediction. Similarly, we'll do the same thing for summaries. So we'll end up with `6,000` total training sentences, `3,000` for the summarize, `3,000` for the sentiment prediction task.
9. Similar to last exercise construct the `Dataset` object from our dataset. The easiest way to create this is using the `Dataset.from_pandas` function.
10. Similarly, as the other exercise, we will pre-process the dataset examples by tokenizing them using our GPT-2 tokenizer.
11. Perform a train test split by randomly sampling `80%` for training, `20%` for testing.

12. Finally, we use the same collator as before, `DataCollatorForLanguageModeling` with our tokenizer. `mlm` set to `false` because we're not performing a mask language model. We're performing an auto aggressive language model.
13. Set up the training arguments such that you go over our dataset twice, batch sizes of 32, load the best model at the end, and evaluate and save our model with checkpoints every epoch.
14. Setup the `Trainer`. Train, evaluate and save your model.
15. Test your model using couple of random examples form the test set.

Part C - Deployment to HuggingFace Spaces

1. Deploy, the model of your choice (from part A or B) to huggingface spaces.
2. Use the API access to your model on huggingface spaces to create a `HTTP` request with a test example.

The task is worth **14 points (5+7+2 points for part A,B,C respectively)** and is due on **March 1 , at 10:00 CET**. For submission, you only need to upload your notebook to ILIAS. Run all cells, and do not clear out the outputs, before submitting.