Deep Learning for NLP 2022
Homework 5

Due on June 27, 2022 at 23:59

Version of June 22, 2022

Overall: 10 points. This homework involves training more complex neural networks which can take some time. **Plan accordingly for the extended training time!**

Hint: You can check out Google Colab for “free” GPU access.

Submission Guidelines

This homework comes with a zip archive containing the data and code templates. Write your code only in the `p1.py` file. Then upload a zip file with the following folder structure:

```
hw5_group###.zip
 ├─hw5.pdf
 └─p1_cpu.py / p1_cuda.py / p1_colab.ipynb
```

- `hw5.pdf` should answer the questions **and** report your results.
- `###` is your group number (e.g., the file name could be `hw5_group2.zip`).
- Please make sure that your code prints **all** relevant results for **all** subtasks when the tutors run `p1.py`.[a]

[a] If you think that it makes your code more readable/structured, you can also submit other `.py` files whose functions are called in `p1.py`. But the tutors will only execute `p1.py`.

1 Sentiment Analysis with Transformers (10P)

In this task, you will utilize a pretrained transformer-based architecture with PyTorch and the Hugging Face library to perform sentiment classification on a movie review dataset. As we will use these two new libraries, you might find these tutorials very helpful:

- Data preparation:
  `https://huggingface.co/docs/transformers/training#prepare-a-dataset`
- Train in native PyTorch:
  `https://huggingface.co/docs/transformers/training#train-in-native-pytorch`

1.1 Update conda environment (0P)

As training transformer-based models is computationally quite expensive, we propose three different settings that you can use to complete this homework. Depending on your individual resources, you can either train the model on your CPU, GPU or in Colab:

CPU-only:
Extend the initial conda environment for this course by following these instructions:

1. Install PyTorch:
   conda install pytorch torchvision torchaudio cpuonly -c pytorch
2. Install Hugging Face Transformers :
   conda install -c huggingface transformers
3. Install Huggingface Datasets:
   pip install datasets

   If you run into import errrors related to the HfApi, the following downgrade should solve the issue.
4. Downgrade huggingface_hub: pip install huggingface_hub==0.4

   (For Apple users, the provided conda environment also works with python=3.7 instead of 3.6)

Nvidia-GPU
Extend the initial conda environment for this course by following these instructions:

1. Install PyTorch:
   conda install pytorch torchvision torchaudio cudatoolkit=11.3 -c pytorch
2. Install Huggingface Transformers:
   conda install -c huggingface transformers
3. Install Huggingface Datasets:
   pip install datasets

   If you run into import errors related to the HfApi, the following downgrade should solve the issue.
4. Downgrade huggingface_hub: pip install huggingface_hub==0.4

Google Colab

Colab has many preinstalled packages, including all from the initial homework conda environment, with slightly different package versions. Therefore, you only need to execute the following line at the start of your Colab notebook (you can either train with or without GPU):

!pip install transformers datasets

1.2 Data and Model Preparation (1P)

The transformer-based model we will use is a variant of the XtremeDistil Transformer by Mukherjee et al. [2021] and you can gain access to it via the identification string **microsoft/xtremedistil-l6-h256-uncased** [1]. Also, the dataset we will use to train the model can be accessed via the identification string **rotten_tomatoes** [2].

Tasks:

1. Set the torch seed to a specific number so your results are reproducible.
2. Load the rotten tomatoes data set .
3. Load the XtremeDistil Transformer as Sequence Classification model with two labels.
4. Load the Tokenizer for the XtremeDistil Transformer.

   Hint: The imports AutoTokenizer and AutoModelForSequenceClassification might be helpful for tasks 3 and 4.

[1] https://huggingface.co/microsoft/xtremedistil-l6-h256-uncased
[2] https://huggingface.co/datasets/rotten_tomatoes

1.3 Preprocessing (4P)

Now that we have loaded the data, the tokenizer and the model, it is time to prepare the data for the model.

Tasks:

1. Preprocess the texts of each data set split (train, validation, test) with the tokenizer, using a max length of **256**, truncation and padding to max length.
2. Shuffle each data set split with a fixed seed of **123** and select **2000** data instances from the train set, and **200** from the validation and test set respectively. Then set the format of each split to torch.
3. Create PyTorch dataloaders for each split, setting the batch size to **8** and setting shuffle to true for the train data loader.

1.4 Model Training (4P)

With the data prepared and ready to load, we can start to train our model.

Tasks:

1. Use Adam as optimizer and set the learning rate to $\mathbf{2 * 10^{-5}}$.

   Hint: `https://huggingface.co/docs/transformers/main_classes/optimizer_schedules#transformers.AdamW`
2. Train the model for **4** epochs on the train set.
3. Track the model loss for each training step. After the training is finished, create a simple loss plot with meaningful axis names and **display it in the pdf**.

   Hint: `https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html`
4. After each epoch, compute the accuracy and the F1 score of the model on the validation set. After the training is finished, plot the accuracy and the F1 score and **display it in the pdf**. (You can either create two separate plots for each metric or display both metrics in one plot.)
5. Save the parameters of your model after each epoch if it beats the best F1 score on the validation set of the previous epochs.

   Hint: `https://huggingface.co/docs/transformers/v4.19.2/en/main_classes/model#transformers.PreTrainedModel.save_pretrained`

1.5 Model Testing (1P)

After training our model, we now need to test the performance of our model on the test set. **Print and report** your accuracy and F1 score on the test set.

References

S. Mukherjee, A. H. Awadallah, and J. Gao. Xtremedistiltransformers: Task transfer for task-agnostic distillation. *CoRR*, abs/2106.04563, 2021. URL `https://arxiv.org/abs/2106.04563`.