Natural Language Processing – Ex2

Due: 8.1.2024 23:59

In this Python exercise, we will experiment with text classification over a subset of the *20newsgroups* dataset. We will compare three different neural models: two versions of Multi-Layer Perceptrons (MLPs) and a pretrained Transformer model.

We provide a skeleton python file *ex2.py*. This is intended as a guideline; you may change it.

Please submit a single zip file. The zip file should contain your code and a pdf file with the results and answers to the last question.

Required Packages: To carry out the exercise, you will need to install the *PyTorch*, *transformers* and *scikit-learn* packages. Additionally, you may use the *evaluate* package.

For installation instructions see:

`https://pytorch.org/get-started/locally/#start-locally`

`https://huggingface.co/docs/transformers/installation`

`https://scikit-learn.org/stable/install.html`

`https://huggingface.co/docs/evaluate/installation`

Notice: You can use the preinstalled *PyTorch* and *sklearn* on the aquarium computers. Also note that if you are not using a GPU you do not need to install CUDA.

Data: We will work with the *20newsgroups* dataset. This dataset comprises around 18000 news posts on 20 topics. In this exercise, we will restrict ourselves to 4 topics: ['comp.graphics', 'rec.sport.baseball', 'sci.electronics', 'talk.politics.guns'].

For additional information see:

`https://scikit-learn.org/stable/datasets/real_world.html#the-20-newsgroups-text-dataset`

Classification Tasks

1. We will run and evaluate a Log-linear classifier. You will implement this as a single layer perceptron and train it with PyTorch.

   For encoding the text you will use TFIDF vectors. Use a feature dimension of 2000 (see the sklearn documentation).

   Remark: Term-Frequency-Inverse-Document-Frequency, or TFIDF, is just a more sophisticated form for a Bag-Of-Words representation. In addition to the normalized term count (TF), we divide by the document frequency (IDF), this gives a penalty to terms that appear in many documents.

   For more information, see:

   `https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html`

   Run classification for 4 different portions of the train data (see the helper function):

   (a) 10 percent of the data (*portion=0.1*)

   (b) 20 percent of the data (*portion=0.2*)

(c) Half the data (*portion=0.5*)

(d) All the data (*portion=1*)

Train for 20 epochs with *learning-rate*= $1e-3$ and *batch-size*= 16. Use the *Adam* optimizer.

Plot (separately) the model loss on the train set and accuracy on the validation set for each epoch. Do this for each proportion of the data (altogether 8 graphs, 2 for each portion).

2. Now we will use an MLP instead of a linear classifier. Define a model with one hidden layer with dimension 500.

   Run the same experiments (with the same settings) as in the previous section. The only change is the model being trained.

3. We will finetune a Transformer model with a SequenceClassification head and an appropriate tokenizer (see skeleton file).

   We will load the pretrained *distilroberta-base* model. Then train the model for 3 epochs with the AdamW optimizer `https://pytorch.org/docs/stable/generated/torch.optim.AdamW.html`, *learning-rate*=5e-5 and *batch-size*=16. This part was done for you in the skeleton file.

   Calculate and report the model's average loss during training (i.e., the losses used by the optimizer) and the accuracy on the validation set, for each epoch.

   In this section, train the model only with 10% and 20% of the data. Plot the train loss and test accuracy for each epoch.

   Notice: We provided helper classes and functions in the skeleton. Notice that the Transformer model is also a PyTorch module so you can train it like any other neural network. Do NOT use the Huggingface Trainer class. Feel free to change the implementation if you wish.

4. Compare the three models:

   (a) Which model had the highest accuracy?

   (b) Which model was the most sensitive to the size of the training set?

   (c) Extract (in your code) and report the number of trainable parameters in each model. Do additional parameters help? Explain.

Good luck!