TÉCNICO LISBOA

Natural Language Lab
Tagus and Alameda
Part 2 – Lab 14

Goals of this lab:
- Solve your project with a pre-trained model
- Identify some important parameters in deep learning architectures

1st STEP: Download the code

git clone https://github.com/ricardorei/lightning-text-classification.git

2nd STEP: Create a virtual environment

virtualenv -p python3.6 venv
source venv/bin/activate

3rd STEP: Install requirements

pip install -r requirements.txt

4th STEP: Read about the code

Main files:

A) tokenizer.py
B) classifer.py
C) training.py

A) ***tokenizer.py***: implements class *Tokenizer*, which defines an interface with Hugging Face tokenizers.

B) ***classifier.py***: defines all the model logic. Follows the structure of a *Lightning Module* (give it a look), meaning that we need to define several methods, which are common to all deep learning models:
- *forward*
- *training_step*
- *validation_step*
- *validation_end*

C) ***training.py***: read terminal commands and initializes the model and pytorch lightning trainer.

Some important arguments:
- batch_size
- accumulate_grad_batches
- min_epochs
- max_epochs
- nr_frozen_epochs (number of epochs before starting to train the pretrained model)
- encoder_learning_rate (learning rate if we decide to update BERT weights)
- learning_rate (classifier learning rate)

5th STEP: Perform the following tests

Test 1: Create a baseline

Train the baseline:
python3 training.py --gpus 0 --batch_size 32 --accumulate_grad_batches 1 --loader_workers 8 --nr_frozen_epochs 1 --encoder_model google/bert_uncased_L-2_H-128_A-2 --train_csv data/MP2_train.csv --dev_csv data/MP2_dev.csv --max_epochs 5

Test the baseline:
python test.py --experiment experiments/version_XXXX --test_data data/MP2_test.csv

Write the information about the command line and the obtained results somewhere.

Test 2: Play with different parameters

Change, for instance, batch size, num of epochs, learning rate. For instance, check nr_frozen_epochs == max_epochs vs. nr_frozen_epochs = 0

Test 3: CLS token

It is normal, as explained in class, that BERT models use the [CLS] token as the sentence embedding. Change the function forward, so that the [CLS] token is used.

Test 4: Test with a new pretrained model

Instead of *google/bert_uncased_L-2_H-128_A-2*, try, for instance, *google/bert_uncased_L-4_H-256_A-4.*
Think a bit about using an encoder with more layers and more attention heads.

Test 5: Add F1 metric (early stoping should change based on F1)

Tip: play with *validation_step* and *validation_end* to add F1. Take a look at pytorch lightning available metrics