

Natural Language Processing – Ex5

Due: 27.1.2023 23:55

In this Python exercise we will experiment with simple Transformer models via the task of text classification over a subset of the *20newsgroups* dataset. We will compare three different models: two common uses of transformers and a simple linear model.

We provide a skeleton python file *ex5.py*. This is intended as a guideline and you may change it if you wish.

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: In order to carry out the exercise, you will need to install the *transformers* and *scikit-learn* packages.

For installation instructions see:

<https://huggingface.co/docs/transformers/installation>

<https://scikit-learn.org/stable/install.html>

Notice: You can use the preinstalled *sklearn* on the aquarium computers.

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. Run and evaluate a Log-linear classifier.

For the classifier you will use a the Logistic Regression model. For encoding the text you will use TFIDF vectors.

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.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

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

Run classification for 3 different portions of the data:

- (a) All data (*portion=1*)
- (b) Half the data (*portion=0.5*)

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

Plot the model accuracy results as a function of the portion of the data.

2. Finetune a Transformer model with a SequenceClassification head and an appropriate tokenizer (see skeleton file).

Use the pretrained *distilroberta-base* model for 5 epochs with *learning-rate=5e-5* and *batch-size=16*. Calculate the model's accuracy.

Notice: We provided some helper classes in the skeleton. See also <https://huggingface.co/docs/transformers/quicktour#trainer-a-pytorch-optimized-training-loop> for more details regarding the Trainer. Notice that the Trainer can also be used to calculate the accuracy via the `evaluate` method.

Repeat the process for the three cases in the previous question.

Again, plot the model accuracy results as a function of the portion of the data.

3. Run zero-shot classification for this task. Use the pretrained '*cross-encoder/nli-MiniLM2-L6-H768*' model with the '*zero-shot-classification*' pipeline.

See https://huggingface.co/docs/transformers/main_classes/pipelines#transformers.ZeroShotClassificationPipeline for more details.

Remark: There are different methods for zero-shot inference. In some cases we can use zero-shot generation. In cases where we limit ourselves to a small set of responses, it is common to separately input all the options into the model, and compare the scores for each input. The predicted output will be that with the highest score.

Report the accuracy you got.

4. Compare the models:

- (a) Which model had the highest accuracy?
- (b) Which model was the most sensitive to the size of the training set?
- (c) Mention 2 pros and 2 cons of the zero-shot model (in comparison to the other models).

Good luck!