Project explanation

We want to do multi-class text classification with transformer models.

Model specifications

To do that we want to test five transformer models from the hugging face library against each other:

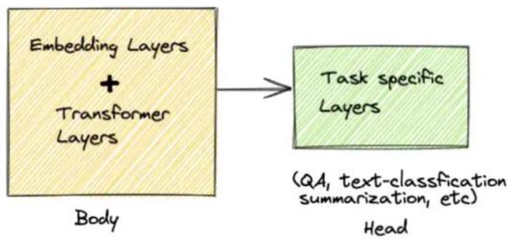
- Bert-base-german-cased
- Dbmdz/bert-base-german-uncased
- Uklfr/gottbert-base
- Deepset/gbert-base
- Xlm-roberta-base

So we need the code for all of the above models.

We want to test the transformer models against each other several times:

1) We want to use the most basic configuration of the transformer model with the most basic way to do the classification (like AutoModelForSequenceClassification, not building our own head)
2) We want to do fine-tuning (see below) and change the classification head like adding custom layers on top of the hugging face model (dropout layer, extract embeddings, dense layer with x units to output x classes) – code must be provided for this

Graphic explanation of what we mean here:



And a blog post to make clear what we mean: https://towardsdatascience.com/adding-custom-layers-on-top-of-a-hugging-face-model-f1ccdfc257bd

Also it is important that you highlight where we need to set the **seeds**, so that we can get the same results!

Fine-tuning options

We want to have the following options for fine-tuning later:

- Give class weights to smaller classes – must be explained in the code where and how to do it
- Change the optimizer (SGD, Adam, Adafactor) – must be explained in the code where and how to do it
- Change the learning rate (different learning rates ranging from 7e-7 to 2e-2) – must be explained in the code where and how to do it
- Change the training and evaluation batch size (8, 16, 32) – must be explained in the code where and how to do it
- Change the number of training epochs (1, 2, 3) – must be explained in the code where and how to do it
- We want to be able to change the sequence length in the model – how to do it must be made clear in the code.

We want to evaluate the models every x steps. Highlight in the code where we can change the number of evaluation steps.

We have a separate heldout dataset for testing later – so please provide the pipeline where we can put in the testset into the final models.

Model output specifications

Metrics

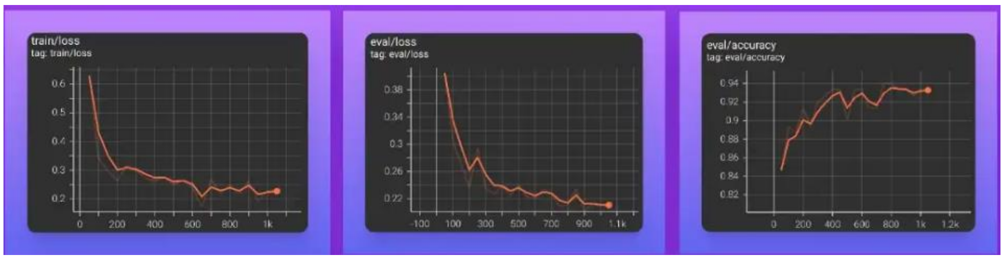
We want to compare the performance of the models based on three metrics:

- MCC (Matthews correlation coefficient)
- Accuracy
- Macro-F1-score

So the output of the model should clearly display those three metrics.

Graphs

We want to check whether the model is overfitting. To do that, we need a graph to each model run that displays the loss and the accuracy every x steps, so we get something like this:



Please provide the code for this.

Attention weight alignment

Also we want to see the attention weight alignment for each model, to observe which tokens the model gave the most attention to. Please provide the code for this.

Wrong predictions

We want to inspect the examples that the model did not predict correctly. Therefore the model must output the wrong classifications with the following information:

- The text
- The correct label
- The predicted label

Preprocessing specifications

We want to split the data into two datasets: training, evaluation. Please include this part in the code as well and highlight where to set the seed, so that we have the same split every time for each model.
The text of the data does not have the same length, so the text must probably be padded and truncated. Please include that in the code.

Also we want to use different datasets with the models – so it must be made clear in the code where we have to adjust the code for a different number of classes later.

Summary

We need the whole text classification pipeline from importing all the necessary libraries to loading and preprocessing/tokenize the data, to the transformer models (and adjustments to it), to the output.

We will be running the code on the following machine:

Google Colab with this environment: NumPy/SciPy/scikit-learn. This is the machine type: 96 vCPUs, 360 GB RAM and this is the GPU: NVIDIA T4 x 4. So the code must run on GPU!