Exercise 7

Ante-hoc Methods

- You can get a bonus point for this exercise if you pass at least 2/3 of the tests. Code is automatically tested after pushing. If your tests fail, either your code is wrong, or you solved the task differently. In the second case, we will manually check your solution.
- Three collected bonus points result in a 0.33 increase of the final grade.
- You are allowed to work in groups with up to three people. You do not need to specify your ErgebnisPIN as it is already collected from the entrance test.
- Follow the ‘README.md‘ for further instructions.
- Finish this exercise by 1st December, 2021 at 11:59 pm.

Note: This assignment requires installing new packages. You can install them by running *pip install -r requirements.txt* in your environment. Additionally, you have to run *python utils/dataset.py* to download and preprocess the dataset.

Instance-wise Feature Selection with Select and Predict Models

Select and predict style models are composed of a selector, that predicts a binary mask over the input features of each instance, and a predictor, that consumes the masked input to make a final prediction. Unfortunately, binary masking is a non-differentiable operation. This makes it hard to train such models end-to-end. A workaround is the so-called *pipeline* setup, where selector and predictor are trained independently. In addition to a label for each instance to train the predictor, this requires groundtruth explanations

In this exercise, the goal is to train and run such a pipeline model on a movie review sentiment classification dataset. The dataset also includes rationale annotations, i.e. highlights that represent important tokens in form of a binary mask. We want to use these as groundtruth explanations to train the selector model. Here are two examples:

text: A gorgeous musical I watched at the palace cinema
label: 1 (positive)
rationale: $[0, 1, 0, 0, 0, 0, 0, 0, 0]$

text: What a bad drama .
label: 0 (negative)
rationale: $[0, 0, 1, 0, 0]$

Note that the 1’s in the rationales highlight the tokens that are important for sentiment classification.

Selector Model

The goal of the selector model is to predict these rationale masks. In our case, the selector is a token classifier, that predicts either 0 or 1 for each token in the sequence. For this exercise, we choose DistilBERT for both the selector and the predictor model. Running the selector consists of the following steps:

1. First, the input text has to be tokenized, that means, the input text tokens are mapped to a sequence of integer input ids. For BERT-style models, a few special tokens are added: Each sequence starts with a [CLS] token and ends with a [SEP] token. Since all sequences in one batch must be of the same

length, a [PAD] token is used to pad shorter sequences to the same length as the longest one in the batch. For a batch containing our two examples, the tokenized input could look like this:

> text: [CLS] A gorgeous musical I watched at the palace cinema [SEP]
> input ids: $[101, 2, 876, 1098, 5, 66, 78, 134, 867, 555, 102]$
> attention_mask: $[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$
> rationale: $[-100, 0, 1, 0, 0, 0, 0, 0, 0, 0, -100]$
>
> text: [CLS] What a bad drama . [SEP] [PAD] [PAD] [PAD] [PAD]
> input ids: $[101, 44, 2, 11, 43, 3, 102, 0, 0, 0, 0]$
> attention_mask: $[1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]$
> rationale: $[-100, 0, 0, 1, 0, 0, -100, -100, -100, -100, -100]$

The special tokens [CLS], [SEP], [PAD] are represented by the special input ids 101, 102 and 0, respectively. Also, the rationale is aligned to the tokenized input ids and is filled with -100 to indicate that the label and the token at that index should be ignored when computing the loss or performance metrics.

2. The model takes as input the input_ids and attention_masks. The output obtained by running a forward pass are logits of shape (batch size, 2, sequence length). These are the unnormalized probabilities for each of the two classes for each token in the input.

3. After obtaining the predictions, the input tokens for which the predicted label is 0, can be masked by replacing the corresponding token ids with the token id of the [MASK] token.

4. Finally, the input ids can be converted back to string text using a decode function, that inverts the tokenization process. Any [CLS], [SEP], [PAD] tokens can be dropped. An example result could be:

   masked text: [MASK] gorgeous [MASK] [MASK] [MASK] [MASK] [MASK] [MASK] [MASK]

Predictor Model

The predictor model is a sequence classification model, as its goal is to predict a single label for every input sequence. Given a movie review text that was masked by the selector model, it predicts either 1 (positive) or 0 (negative). The inputs must be tokenized in the same way as for the selector. The model output is now of shape (batch size, 2). Again, these are the logits for each class.

1 Select and Predict

1.1 Training the Selector Model

In this exercise, we build on pretrained models in this exercise. Yet, the selector needs some finetuning to the new task of extracting explanation masks. Complete the *train_selector_model* function that trains the token classification head of the DistilBERT model **for one epoch**. This includes implementing the sub-function *train_one_epoch* as well as *validate*, that evaluates the model on the validation set after training it. Use cross entropy loss and make sure to not include the tokens for which the rationale label is -100. For more details, refer to the in code documentation.

1.2 Implementing the Pipeline Model

Complete the functions *select* and *predict*, that implement the two steps of the select-and-predict pipeline. Associated file: *select_and_predict.py.*