Baseline: Fine-tuning language model

Firstly, I tried to find the best and fastest way to solve the task. With this in mind, I went to the Hugging Face platform and found the directory with the paraphraser (https://huggingface.co/s-nlp/t5-paranmt-detox) of the team, from whose Github directory I downloaded the raw dataset for this assignment (https://github.com/s-nlp/detox). I noticed that their paraphraser is, in fact, based on the ceshine/t5-paraphrase-paws-msrp-opinosis model. So, I decided to fine-tune it for my task and conduct several experiments to find out the best performing training method.

Experiments were carried out on the Google Collab platform, since my PC did not have a GPU unit, without which it was impossible to train and test the model at the adequate time.

First experiment: Training the ceshine/t5-paraphrase-paws-msrp-opinosis model with "turn toxic to neutral" prefix for the input data.

I thought it would be a great idea to add a prefix that corresponds to the task I tried to solve with the model from the T5 family. Evaluation metric: *bleu score*. As a result of training using part of the preprocessed data, I got the following results:

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	No log	1.530098	22.977900	14.202000
2	No log	1.474448	23.728800	14.050000
3	No log	1.449903	24.224800	14.004000
4	1.647400	1.437334	24.521100	14.004000
5	1.647400	1.423515	24.511700	13.954000
6	1.647400	1.420065	24.634600	13.960000
7	1.485900	1.414913	24.969400	13.942000
8	1.485900	1.412552	24.907500	13.894000
9	1.485900	1.411969	25.064900	13.934000
10	1.431600	1.411508	25.106500	13.942000

```
TrainOutput(global_step=1570, training_loss=1.5165174350617039, metrics={'train_runtime': 699.6884, 'train_samples_per_second': 71.46,
'train_steps_per_second': 2.244, 'total_flos': 3068776684707840.0, 'train_loss': 1.5165174350617039, 'epoch': 10.0})
```

Trained model showed an acceptable level of paraphrasing performance. (See *1.0_initial_train_and_test.ipynb* notebook for the full code)

Hypothesis 1: Train without prefix

However, after reading some articles and discussions about prefixes for T5 models, I became unsure if that prefix I used previously was useful for the detoxification type of a task: it may have not been used in the initial training of the T5 model.

Because of that, I tried to train the ceshine/t5-paraphrase-paws-msrp-opinosis model without any prefix attached to input data. Here is the training log:

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	No log	1.557626	21.686000	13.978000
2	No log	1.487002	23.025000	14.076000
3	No log	1.456288	24.129800	14.040000
4	1.707800	1.440308	24.290800	14.006000
5	1.707800	1.427558	24.268900	14.030000
6	1.707800	1.422584	24.621200	13.980000
7	1.488700	1.416779	24.795100	13.990000
8	1.488700	1.414756	24.863000	13.994000
9	1.488700	1.413583	24.864800	14.006000
10	1.427400	1.413186	24.819000	13.998000

```
TrainOutput(global_step=1570, training_loss=1.53495962543852, metrics={'train_runtime': 691.8754, 'train_samples_per_second': 72.267, 'train
2830655119564800.0, 'train_loss': 1.53495962543852, 'epoch': 10.0})
```

Compared to the results of previous experiments, I noticed that training gave slightly worse results with approximately the same execution time. In conclusion, using a prefix is better. (See *1.1_train_and_test_without_prefix.ipynb* notebook for full code)

Hypothesis 2: Train with better arguments

After that experiments with prefixes, I wanted to try to improve model performance by using better training arguments.

To do this, I researched the Hugging Face docs and decided to change some parameters, changing them to values that were closer to the default ones. These changes have led me to better results:

Epoch	Training Loss	Validation Loss	Bleu	Gen Len
1	No log	1.456775	24.059100	14.080000
2	No log	1.413215	24.885400	13.936000
3	No log	1.397439	25.147800	13.924000
4	1.567100	1.392835	25.193300	13.870000
5	1.567100	1.389202	25.420500	13.886000
6	1.567100	1.393260	25.513300	13.822000
7	1.310600	1.392417	25.337400	13.824000
8	1.310600	1.394696	25.874200	13.796000
9	1.310600	1.395465	25.785900	13.782000
10	1.219100	1.397373	25.687000	13.768000

```
TrainOutput(global_step=1570, training_loss=1.3576157247944243, metrics={'train_runtime': 696.2629, 'train_samples_per_second': 71.812,
'train_steps_per_second': 2.255, 'total_flos': 3068776684707840.0, 'train_loss': 1.3576157247944243, 'epoch': 10.0})
```

In short, all indicators have improved. (See *1.2_better_training_params.ipynb* notebook for full code)

Results:

In conclusion, I stopped at using ceshine/t5-paraphrase-paws-msrp-opinosis model fine-tuned with the use of prefixes and optimal training parameters as my final solution.