fine_tuning-3

May 1, 2024

Main idea: In this project, we will use Twitter-roBERTa-base for Sentiment Analysis. In other words, we will train the model which is able to evaluate if the text on Twitter is positive, negative, or neural.
Data: https://huggingface.co/datasets/tweet_eval
Model: https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment-latest

[1]:
```
!pip install accelerate -U
```

```
Requirement already satisfied: accelerate in /usr/local/lib/python3.10/dist-
packages (0.28.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from accelerate) (1.25.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from accelerate) (23.2)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages
(from accelerate) (5.9.5)
Requirement already satisfied: pyyaml in /usr/local/lib/python3.10/dist-packages
(from accelerate) (6.0.1)
Requirement already satisfied: torch>=1.10.0 in /usr/local/lib/python3.10/dist-
packages (from accelerate) (2.1.0+cu121)
Requirement already satisfied: huggingface-hub in
/usr/local/lib/python3.10/dist-packages (from accelerate) (0.20.3)
Requirement already satisfied: safetensors>=0.3.1 in
/usr/local/lib/python3.10/dist-packages (from accelerate) (0.4.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->accelerate) (3.13.1)
Requirement already satisfied: typing-extensions in
/usr/local/lib/python3.10/dist-packages (from torch>=1.10.0->accelerate)
(4.10.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->accelerate) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->accelerate) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->accelerate) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages
(from torch>=1.10.0->accelerate) (2023.6.0)
Requirement already satisfied: triton==2.1.0 in /usr/local/lib/python3.10/dist-
packages (from torch>=1.10.0->accelerate) (2.1.0)
```

```
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub->accelerate) (2.31.0)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-
packages (from huggingface-hub->accelerate) (4.66.2)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.10.0->accelerate)
(2.1.5)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub->accelerate) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->huggingface-hub->accelerate) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub->accelerate) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->huggingface-
hub->accelerate) (2024.2.2)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-
packages (from sympy->torch>=1.10.0->accelerate) (1.3.0)
```

[2]:
```
!pip install transformers datasets
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-
packages (4.38.2)
Requirement already satisfied: datasets in /usr/local/lib/python3.10/dist-
packages (2.18.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-
packages (from transformers) (3.13.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.19.3 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.20.3)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from transformers) (1.25.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from transformers) (23.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-
packages (from transformers) (6.0.1)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.10/dist-packages (from transformers) (2023.12.25)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-
packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers<0.19,>=0.14 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.15.2)
Requirement already satisfied: safetensors>=0.4.1 in
/usr/local/lib/python3.10/dist-packages (from transformers) (0.4.2)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-
packages (from transformers) (4.66.2)
Requirement already satisfied: pyarrow>=12.0.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from datasets) (14.0.2)
Requirement already satisfied: pyarrow-hotfix in /usr/local/lib/python3.10/dist-
packages (from datasets) (0.6)
Requirement already satisfied: dill<0.3.9,>=0.3.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from datasets) (1.5.3)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages
(from datasets) (3.4.1)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dist-
packages (from datasets) (0.70.16)
Requirement already satisfied: fsspec[http]<=2024.2.0,>=2023.1.0 in
/usr/local/lib/python3.10/dist-packages (from datasets) (2023.6.0)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.10/dist-
packages (from datasets) (3.9.3)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.3.1)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (23.2.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (1.4.1)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (6.0.5)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-
packages (from aiohttp->datasets) (1.9.4)
Requirement already satisfied: async-timeout<5.0,>=4.0 in
/usr/local/lib/python3.10/dist-packages (from aiohttp->datasets) (4.0.3)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.10/dist-packages (from huggingface-
hub<1.0,>=0.19.3->transformers) (4.10.0)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-
packages (from requests->transformers) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests->transformers) (2024.2.2)
Requirement already satisfied: python-dateutil>=2.8.1 in
/usr/local/lib/python3.10/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas->datasets) (2023.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.1->pandas->datasets) (1.16.0)
```

[4]:
```
from transformers import AutoModelForSequenceClassification, AutoTokenizer,␣
↪Trainer, TrainingArguments, set_seed
```

```
from sklearn.metrics import classification_report
import datasets
import torch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[5]:
```
LR = 2e-5
EPOCHS = 5
BATCH_SIZE = 32
MODEL = "cardiffnlp/twitter-roberta-base-sentiment-latest"
MAX_TRAINING_EXAMPLES = -1

# set transformers seed
seed = 42
set_seed(seed)
```

0.0.1 LOAD THE DATA

[6]:
```
#Dataset
dataset = datasets.load_dataset("tweet_eval", "sentiment")

# use model's tokenizer to get text encodings
tokenizer = AutoTokenizer.from_pretrained(MODEL, use_fast=True)

dataset = dataset.map(lambda e: tokenizer(e["text"], truncation=True),␣
 ↪batched=True)

# make sure to use whole train dataset if MAX_TRAINING_EXAMPLES == -1
if MAX_TRAINING_EXAMPLES == -1: MAX_TRAINING_EXAMPLES = dataset["train"].
 ↪shape[0]

# split into train/val/test sets
train_dataset = dataset["train"]
val_dataset = dataset["validation"]
test_dataset = dataset["test"]
```

```
Map:   0%|          | 0/45615 [00:00<?, ? examples/s]
```

```
Asking to truncate to max_length but no maximum length is provided and the model
has no predefined maximum length. Default to no truncation.
```

```
Map:   0%|          | 0/12284 [00:00<?, ? examples/s]
```

```
Map:   0%|          | 0/2000 [00:00<?, ? examples/s]
```

Visualize the data set

[7]:
```
dataset
```

[7]:
```
DatasetDict({
    train: Dataset({
        features: ['text', 'label', 'input_ids', 'attention_mask'],
        num_rows: 45615
    })
    test: Dataset({
        features: ['text', 'label', 'input_ids', 'attention_mask'],
        num_rows: 12284
    })
    validation: Dataset({
        features: ['text', 'label', 'input_ids', 'attention_mask'],
        num_rows: 2000
    })
})
```

Understanding the data: Labels: 0 -> Negative; 1 -> Neutral; 2 -> Positive

[6]:
```
print(dataset["train"])
print(dataset["train"][0])
print(dataset["test"])
print(dataset["validation"])
```

```
Dataset({
    features: ['text', 'label', 'input_ids', 'attention_mask'],
    num_rows: 45615
})
{'text': '"QT @user In the original draft of the 7th book, Remus Lupin survived
the Battle of Hogwarts. #HappyBirthdayRemusLupin"', 'label': 2, 'input_ids': [0,
113, 1864, 565, 787, 12105, 96, 5, 1461, 2479, 9, 5, 262, 212, 1040, 6, 8022,
687, 26110, 179, 5601, 5, 9846, 9, 42210, 4, 849, 21136, 44728, 1208, 31157,
687, 574, 658, 179, 113, 2], 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]}
Dataset({
    features: ['text', 'label', 'input_ids', 'attention_mask'],
    num_rows: 12284
})
Dataset({
    features: ['text', 'label', 'input_ids', 'attention_mask'],
    num_rows: 2000
})
```

0.0.2 PREPROCESS THE DATA USING TOKENIZER

[8]:
```
from transformers import AutoTokenizer
import torch

tokenizer = AutoTokenizer.from_pretrained(MODEL)

# Define a maximum sequence length for padding/truncation
```

```
max_length = 64

def tokenize_and_encode(examples):
    # Tokenize the text
    tokenized_inputs = tokenizer(examples["text"], truncation=True,␣
 ↪padding='max_length', max_length=max_length, return_tensors='pt')
    tokenized_inputs["label"] = examples["label"]
    return tokenized_inputs

train_dataset = train_dataset.map(tokenize_and_encode, batched=True)
val_dataset = val_dataset.map(tokenize_and_encode, batched=True)
test_dataset = test_dataset.map(tokenize_and_encode, batched=True)
```

```
Map:   0%|          | 0/2000 [00:00<?, ? examples/s]
```

0.0.3 TRAINING THE MODEL USING TRAINER API

[7]:
```
training_args = TrainingArguments(
    output_dir="./results",                    # output directory
    num_train_epochs=EPOCHS,                   # total number of training epochs
    per_device_train_batch_size=128,           # batch size per device during␣
 ↪training
    per_device_eval_batch_size=128,            # batch size for evaluation
    logging_dir='./logs',                      # directory for storing logs
    logging_steps=160,                         # when to print log
    evaluation_strategy='steps',               # evaluate every n number of␣
 ↪steps.
    load_best_model_at_end=True,               # to load or not the best model␣
 ↪at the end
    save_steps=160,                            # create a checkpoint every time␣
 ↪we evaluate,
    seed=seed                                  # seed for consistent results
)

model = AutoModelForSequenceClassification.from_pretrained(MODEL, num_labels=3)
```

```
/usr/local/lib/python3.10/dist-packages/torch/_utils.py:831: UserWarning:
TypedStorage is deprecated. It will be removed in the future and UntypedStorage
will be the only storage class. This should only matter to you if you are using
storages directly.  To access UntypedStorage directly, use
tensor.untyped_storage() instead of tensor.storage()
  return self.fget.__get__(instance, owner)()
Some weights of the model checkpoint at cardiffnlp/twitter-roberta-base-
sentiment-latest were not used when initializing
RobertaForSequenceClassification: ['roberta.pooler.dense.bias',
'roberta.pooler.dense.weight']
- This IS expected if you are initializing RobertaForSequenceClassification from
the checkpoint of a model trained on another task or with another architecture
```

```
(e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing RobertaForSequenceClassification
from the checkpoint of a model that you expect to be exactly identical
(initializing a BertForSequenceClassification model from a
BertForSequenceClassification model).
```

Now, lets change the model and see how the loss fucntion performs.

[9]:
```
trainer = Trainer(
    model=model,
    tokenizer=tokenizer,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset
)

trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/accelerate/accelerator.py:432:
FutureWarning: Passing the following arguments to `Accelerator` is deprecated
and will be removed in version 1.0 of Accelerate: dict_keys(['dispatch_batches',
'split_batches', 'even_batches', 'use_seedable_sampler']). Please pass an
`accelerate.DataLoaderConfiguration` instead:
dataloader_config = DataLoaderConfiguration(dispatch_batches=None,
split_batches=False, even_batches=True, use_seedable_sampler=True)
  warnings.warn(
```

```
<IPython.core.display.HTML object>
```

```
Checkpoint destination directory ./results/checkpoint-160 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-320 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-480 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-640 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-800 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-960 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-1120 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-1280 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-1440 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
Checkpoint destination directory ./results/checkpoint-1600 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
```

```
Checkpoint destination directory ./results/checkpoint-1760 already exists and is
non-empty. Saving will proceed but saved results may be invalid.
```

[9]:
```
TrainOutput(global_step=1785, training_loss=0.3216486841356721,
metrics={'train_runtime': 774.1244, 'train_samples_per_second': 294.623,
'train_steps_per_second': 2.306, 'total_flos': 7501199093539200.0, 'train_loss':
0.3216486841356721, 'epoch': 5.0})
```

With the number of epoch of 5, we have the last loss score is 0.121600. However, the validation loss score is 0.942294, which is very high for loss fucntion. It indicates that the model doesn't perform goo on validation (unseen) dataset. Therefore, the data is overfitting.

[ ]:
```
```

[10]:
```
trainer.evaluate()
```

```
<IPython.core.display.HTML object>
```

[10]:
```
{'eval_loss': 0.544927716255188,
 'eval_runtime': 2.0164,
 'eval_samples_per_second': 991.878,
 'eval_steps_per_second': 7.935,
 'epoch': 5.0}
```

[11]:
```
# for every prediction the model ouptuts logits where largest value indicates␣
 ↪the predicted class
test_preds_raw, test_labels , _ = trainer.predict(test_dataset)
test_preds = np.argmax(test_preds_raw, axis=-1)
print(classification_report(test_labels, test_preds, digits=3))
```

```
<IPython.core.display.HTML object>
```

	precision	recall	f1-score	support
0	0.686	0.845	0.757	3972
1	0.772	0.644	0.702	5937
2	0.712	0.731	0.721	2375
accuracy			0.726	12284
macro avg	0.723	0.740	0.727	12284
weighted avg	0.733	0.726	0.724	12284

[12]:
```
from scipy.special import softmax

scores = softmax(test_preds_raw, axis=1)
scores
```

[12]:
```
array([[0.81319547, 0.18189639, 0.00490818],
       [0.07632931, 0.8428638 , 0.08080684],
       [0.40040976, 0.5853274 , 0.01426285],
       …,
       [0.50806063, 0.4868901 , 0.00504932],
       [0.9707836 , 0.02526369, 0.00395279],
       [0.0198051 , 0.12303532, 0.8571596 ]], dtype=float32)
```

Note: Based on your test data, this array shows the probability of each class as predicted by your model. The softmax function is applied to your model's raw output in order to calculate the probability.

A sample from your test data corresponds to each row in the array. Every column represents a class. The model predicts, for instance, that the first sample in your test data will belong to the first class with a probability of 0.81319547, the second class with a probability of 0.18189639, and the third class with a probability of 0.00490818. This is indicated by the first row [0.81319547, 0.18189639, 0.00490818]

Conclusion: when the model is tested on the test data, the accuracy is about 0.726 and the test loss is about 0.545. In particular, the loss fucntion is still too much higher than loss fucntion on the train dataset. This suggest that the dataset is definitely overfitting. However, we got 72.6% as a accuracy, which is not a bad accuracy.

[ ]: