Aim

1. Test one trained model of the HuggingFace library.
2. FineTune, one of the HuggingFace models, show the result.

Analysis

Sentiment analysis is a natural language processing technique that identifies the polarity of a given text. There are different flavors of sentiment analysis, but one of the most widely used techniques labels data into positive, negative and neutral.

There are more than 215 sentiment analysis models publicly available on the HuggingFace Hub and integrating them with Python just takes few lines of code.

```
specific_model = pipeline(model="finiteautomata/bertweet-base-sentiment-analysis")
data = ["I love you", "I hate you"]
specific_model(data)
```

Results are obvious. We got ~0.99 for positive and ~0.979 for negative statement. Model is working. Now we can experiment with more specific phrases.

Specifying ‘white man’ as an input to the model, we get negative score of 0.68, while ‘black man’ is 78.2 % neutral. It can be a result of an often racist context near that collocation, about ‘white people dominance’. ‘nazi’ as input word gets ~0.59 neutral in comparison of ~0.68 negative by ‘neo-nazi’, we can propose that the first word relate only to historical phenomenon, but other one occurs in the articles/news more often after some hate speech or illegal actions.

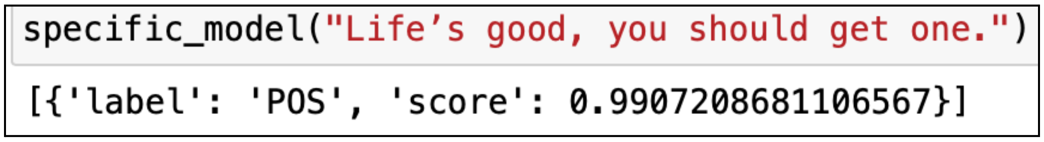
With the help of the next example, we can see how different adjectives and nouns can vary the output of our model.

'killer': '**NEG** 0.867',
'perfect killer': '**POS** 0.643',
'perfect killer bad': '**NEG** 0.942',

'perfect awesome killer bad': **'POS** 0.595',
'perfect awesome killer bad idiot': '**NEG** 0.975',
'perfect awesome brilliant fascinating charming very good the best killer bad idiot': **'POS** 0.98'

It is interesting that the model's evaluation of Adolf Hitler is fully negative while Joseph Stalin is 90% neutral. A word 'holocaust' is neutral until it has an article 'the' (then negative).

The next image shows that our neural network can not understand sarcasm sentences.

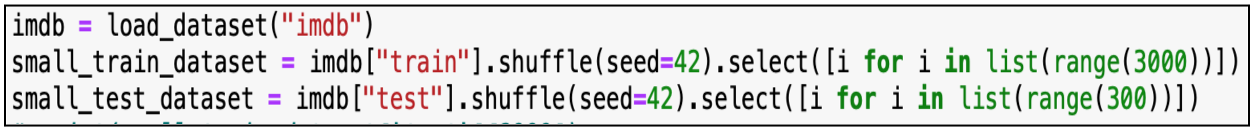


```
specific_model("Life's good, you should get one.")
[{'label': 'POS', 'score': 0.9907208681106567}]
```

Fine tuning

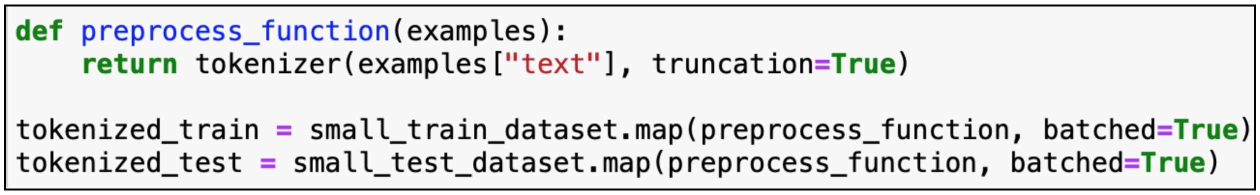
We will use the IMBD dataset to fine-tune a DistilBERT model for sentiment analysis. The IMDB dataset contains 25,000 movie reviews labeled by sentiment for training a model and 25,000 movie reviews for testing it. DistilBERT is a smaller, faster and cheaper version of BERT. It has 40% smaller than BERT and runs 60% faster while preserving over 95% of BERT's performance. We'll use the IMDB dataset to fine-tune a DistilBERT model that is able to classify whether a movie review is positive or negative.

Firstly we install all the dependencies. After that we import our dataset and create a few smaller ones.



```
imdb = load_dataset("imdb")
small_train_dataset = imdb["train"].shuffle(seed=42).select([i for i in list(range(3000))])
small_test_dataset = imdb["test"].shuffle(seed=42).select([i for i in list(range(300))])
```

To preprocess our data, you will use the DistilBERT tokenizer.
tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")



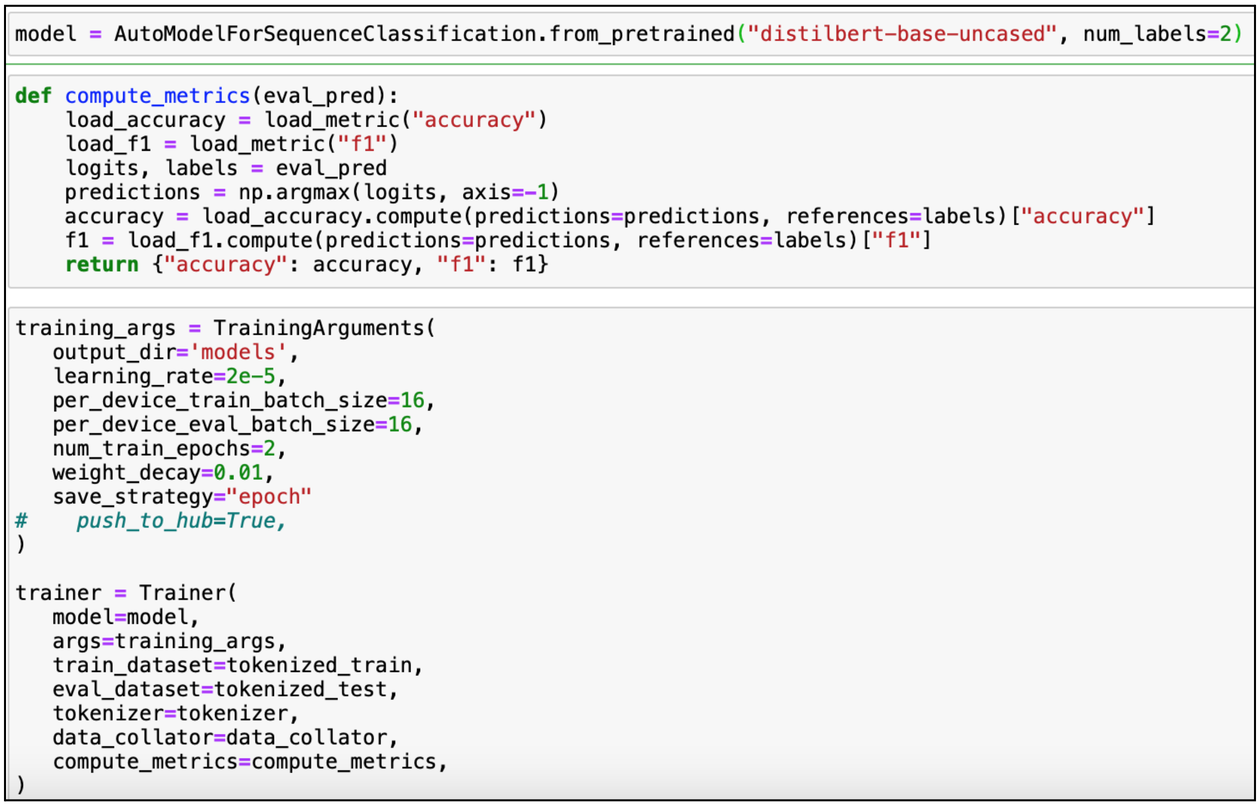
```python
def preprocess_function(examples):
    return tokenizer(examples["text"], truncation=True)

tokenized_train = small_train_dataset.map(preprocess_function, batched=True)
tokenized_test = small_test_dataset.map(preprocess_function, batched=True)
```

To speed up training, let's use a data_collator to convert training samples to PyTorch tensors and concatenate them with the correct amount of padding.

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

We define our model metrics function, ‘trainer object’:



```python
model = AutoModelForSequenceClassification.from_pretrained("distilbert-base-uncased", num_labels=2)

def compute_metrics(eval_pred):
    load_accuracy = load_metric("accuracy")
    load_f1 = load_metric("f1")
    logits, labels = eval_pred
    predictions = np.argmax(logits, axis=-1)
    accuracy = load_accuracy.compute(predictions=predictions, references=labels)["accuracy"]
    f1 = load_f1.compute(predictions=predictions, references=labels)["f1"]
    return {"accuracy": accuracy, "f1": f1}

training_args = TrainingArguments(
   output_dir='models',
   learning_rate=2e-5,
   per_device_train_batch_size=16,
   per_device_eval_batch_size=16,
   num_train_epochs=2,
   weight_decay=0.01,
   save_strategy="epoch"
#    push_to_hub=True,
)

trainer = Trainer(
   model=model,
   args=training_args,
   train_dataset=tokenized_train,
   eval_dataset=tokenized_test,
   tokenizer=tokenizer,
   data_collator=data_collator,
   compute_metrics=compute_metrics,
)
```

After executing ‘trainer.train()’, we have an output:

training_loss: 0.288,
train_samples_per_second: 0.348,

We can mention that the training was extremely slow, which is proofed by the ‘train_samples_per_second’ variable. Loss is ~0.28 that is quite a good index.

In our case after an evaluation, we got 84% accuracy and 86% f1 score. Well done for a model trained just with 3,000 samples. We’ve collected some real reviews from theme web sites.

- 'This fly-on-the-wall rock doc details the original line-up\'s chaotic journey': **POS** 84%,
- 'No one turned up for this gay romcom\'s US run, but Britain\'s response should be better' **NEG** 66%,
- 'Not quite as many thrills, but still fun to dive into' **POS** 79%,
- 'I\'m not sure what accomplished director\/producer\/cinematographer Joshua Caldwell was thinking taking on this project.' **NEG** 87%,
- 'This film has got to be the epitome of terrible writing and should be a classroom example of \'what not to do\' when writing a screenplay. **NEG** 91%'
- ‘Well. What did I just watch ? I\'m not sure if I can find words to describe this movie’ **NEG** 95%.

Testing those sentences we got the following results, which are amazing as I think! A model does not divide things into black and white, it determines intermediate moods/opinions, evaluates them really well.

Nowadays transformers are significantly important in Machine Learning. They are a solution to various business problems such as image, text or audio analysis. In addition to that, those models are really easy to train and use, especially with the help of the transformers module from HuggingFace. With fine-tuned distil-bert today, the model can evaluate an emotion/sentimental state of film reviews.