lab07

June 7, 2023

[ ]:
```
%pip install transformers[sentencepiece]
%pip install datasets
%pip install --upgrade accelerate
%pip install evaluate
```

Let's define all the imports we will need

[3]:
```
import torch
import numpy as np
import evaluate

from datasets import load_dataset, load_metric
from transformers import (
    AutoTokenizer,
    AutoModelForSequenceClassification,
    Trainer,
    TrainingArguments,
    DataCollatorWithPadding,
)
```

[4]:
```
dataset = load_dataset("imdb")
```

```
Downloading builder script:   0%|          | 0.00/4.31k [00:00<?, ?B/s]

Downloading metadata:   0%|          | 0.00/2.17k [00:00<?, ?B/s]

Downloading readme:   0%|          | 0.00/7.59k [00:00<?, ?B/s]

Downloading and preparing dataset imdb/plain_text to /root/.cache/huggingface/da
tasets/imdb/plain_text/1.0.0/d613c88cf8fa3bab83b4ded3713f1f74830d1100e171db75bbd
db80b3345c9c0…

Downloading data:   0%|          | 0.00/84.1M [00:00<?, ?B/s]

Generating train split:   0%|          | 0/25000 [00:00<?, ? examples/s]

Generating test split:   0%|          | 0/25000 [00:00<?, ? examples/s]

Generating unsupervised split:   0%|          | 0/50000 [00:00<?, ? examples/s]

Dataset imdb downloaded and prepared to /root/.cache/huggingface/datasets/imdb/p
lain_text/1.0.0/d613c88cf8fa3bab83b4ded3713f1f74830d1100e171db75bbddb80b3345c9c0
. Subsequent calls will reuse this data.
```

```
  0%|          | 0/3 [00:00<?, ?it/s]
```

Use the HuggingFace transformer library to fine-tune a model on the IMDB library dataset and then evaluate it on the test set.

[ ]:
```
checkpoint = "distilbert-base-uncased"
model = AutoModelForSequenceClassification.from_pretrained(checkpoint,␣
 ↪num_labels=2)
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
```

```
Downloading (…)lve/main/config.json:   0%|          | 0.00/483 [00:00<?, ?B/s]
Downloading pytorch_model.bin:   0%|          | 0.00/268M [00:00<?, ?B/s]
Some weights of the model checkpoint at distilbert-base-uncased were not used
when initializing DistilBertForSequenceClassification:
['vocab_transform.weight', 'vocab_layer_norm.weight', 'vocab_transform.bias',
'vocab_projector.bias', 'vocab_layer_norm.bias', 'vocab_projector.weight']
- This IS expected if you are initializing DistilBertForSequenceClassification
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
DistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some weights of DistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['classifier.bias', 'classifier.weight', 'pre_classifier.weight',
'pre_classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
Downloading (…)okenizer_config.json:   0%|          | 0.00/28.0 [00:00<?, ?B/s]
Downloading (…)solve/main/vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
```

[10]:
```
device = torch.device("cuda") if torch.cuda.is_available() else torch.
 ↪device("cpu")
model.to(device)
device
```

[10]:
```
device(type='cuda')
```

[11]:
```
def tokenize(batch: dict) -> dict:
    """
    Tokenizes a batch of text using the provided tokenizer.

    Args:
```

```
        batch (dict): A dictionary containing the text to be tokenized.

    Returns:
        dict: A dictionary containing the tokenized text.

    """
    return tokenizer(batch["text"], truncation=True)


dataset.set_format("torch", columns=["text", "label"])
dataset = dataset.map(tokenize, batched=True)
dataset.set_format("torch", columns=["input_ids", "attention_mask", "label"])
```

```
Map:   0%|          | 0/25000 [00:00<?, ? examples/s]
Map:   0%|          | 0/25000 [00:00<?, ? examples/s]
Map:   0%|          | 0/50000 [00:00<?, ? examples/s]
```

We will use the Data Collactor object from HuggingFace to batch the data and prepare it for the model.

[12]:
```
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

To compute the metrics from the training, we will use Glue and MRPC from the datasets library.

[ ]:
```
def compute_metrics(eval_preds: tuple) -> dict:
        """
    Computes evaluation metrics for a given model's predictions.

    Args:
        eval_preds (tuple): A tuple containing the logits and labels for␣
 ↪evaluation.

    Returns:
        dict: A dictionary containing the computed evaluation metrics.
    """
    metric = evaluate.load("glue", "mrpc")
    logits, labels = eval_preds
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)
```

Here, a Trainer object is instantiated with various arguments.

- model is the pre-trained model that will be trained.
- args is the TrainingArguments object we created earlier, specifying the training configuration.
- train_dataset the training dataset.
- eval_dataset the evaluation dataset.
- data_collator is an optional object used for custom data collation during training. It can be used to preprocess the input data.
- tokenizer is the tokenizer object used for tokenizing the input data.

- compute_metrics is a function that computes evaluation metrics for the model's predictions.

[ ]:
```
training_args = TrainingArguments("test_trainer", evaluation_strategy="epoch")
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset["train"],
    eval_dataset=dataset["test"],
    data_collator=data_collator,
    tokenizer=tokenizer,
    compute_metrics=compute_metrics,
)
```

[ ]:
```
trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/transformers/optimization.py:407:
FutureWarning: This implementation of AdamW is deprecated and will be removed in
a future version. Use the PyTorch implementation torch.optim.AdamW instead, or
set `no_deprecation_warning=True` to disable this warning
  warnings.warn(
You're using a DistilBertTokenizerFast tokenizer. Please note that with a fast
tokenizer, using the `__call__` method is faster than using a method to encode
the text followed by a call to the `pad` method to get a padded encoding.
```

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

```
Downloading builder script:   0%|          | 0.00/5.75k [00:00<?, ?B/s]
```

[ ]:
```
TrainOutput(global_step=9375, training_loss=0.18653556477864583,
metrics={'train_runtime': 4820.9016, 'train_samples_per_second': 15.557,
'train_steps_per_second': 1.945, 'total_flos': 9363658844900448.0, 'train_loss':
0.18653556477864583, 'epoch': 3.0})
```

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

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

[ ]:
```
{'eval_loss': 0.3719950318336487,
 'eval_accuracy': 0.93032,
 'eval_f1': 0.9304979253112033,
 'eval_runtime': 391.5794,
 'eval_samples_per_second': 63.844,
 'eval_steps_per_second': 7.981,
 'epoch': 3.0}
```

The accuracy on the test set is 0.93032 and the F1 score is 0.93049.

Evaluate the model in term of accuracy on the test data.

[5]:
```
tokenizer = AutoTokenizer.from_pretrained("mvonwyl/
 ↪distilbert-base-uncased-imdb")
```

```
model = AutoModelForSequenceClassification.from_pretrained(
    "mvonwyl/distilbert-base-uncased-imdb"
)
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/360 [00:00<?, ?B/s]
Downloading (…)solve/main/vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
Downloading (…)/main/tokenizer.json:   0%|          | 0.00/711k [00:00<?, ?B/s]
Downloading (…)cial_tokens_map.json:   0%|          | 0.00/125 [00:00<?, ?B/s]
Downloading (…)lve/main/config.json:   0%|          | 0.00/615 [00:00<?, ?B/s]
Downloading pytorch_model.bin:   0%|          | 0.00/268M [00:00<?, ?B/s]
```

[13]:
```
def compute_metrics_accuracy(eval_pred: tuple) -> float:
    """
    Computes the accuracy metric for a given model's predictions.

    Args:
        eval_pred (tuple): A tuple containing the predictions and labels for␣
 ↪evaluation.

    Returns:
        float: The computed accuracy.

    """
    metric = load_metric("accuracy")
    predictions, labels = eval_pred
    predictions = np.argmax(predictions, axis=1)
    return metric.compute(predictions=predictions, references=labels)
```

[16]:
```
training_args = TrainingArguments("test_trainer")
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset["train"],
    eval_dataset=dataset["test"],
    compute_metrics=compute_metrics_accuracy,
    data_collator=data_collator,
)
```

We don't need to train the model this time because we can load the model from HugoingFace's model hub.

[17]:
```
trainer.evaluate()
```

```
You're using a DistilBertTokenizerFast tokenizer. Please note that with a fast
tokenizer, using the `__call__` method is faster than using a method to encode
the text followed by a call to the `pad` method to get a padded encoding.
```

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

```
<ipython-input-13-9d1e27752faf>:2: FutureWarning: load_metric is deprecated and
will be removed in the next major version of datasets. Use 'evaluate.load'
instead, from the new library  Evaluate: https://huggingface.co/docs/evaluate
  metric = load_metric('accuracy')
```

```
Downloading builder script:   0%|          | 0.00/1.65k [00:00<?, ?B/s]
```

[17]:
```
{'eval_loss': 0.3972010016441345,
 'eval_accuracy': 0.92948,
 'eval_runtime': 390.8253,
 'eval_samples_per_second': 63.967,
 'eval_steps_per_second': 7.996}
```

The accuracy is 0.92948 on the test set, which is very good

For at least 2 samples which have been wrongly classified in the test set, try explaining why the model could have been wrong.

[ ]:
```
predictions = trainer.predict(dataset["test"])
predictions = predictions.predictions.argmax(-1)
```

[30]:
```
wrong_predictions = []
for i in range(len(predictions)):
    if predictions[i] != dataset["test"][i]["label"]:
        tokens = tokenizer.
 ↪convert_ids_to_tokens(dataset["test"][i]["input_ids"])
        wrong_predictions.append(
            [dataset["test"][i]["label"], tokenizer.
 ↪convert_tokens_to_string(tokens)]
        )
```

[31]:
```
print(wrong_predictions[0])
print(wrong_predictions[1])
```

```
[tensor(0), "[CLS] first off let me say, if you haven ' t enjoyed a van damme
movie since bloodsport, you probably will not like this movie. most of these
movies may not have the best plots or best actors but i enjoy these kinds of
movies for what they are. this movie is much better than any of the movies the
other action guys ( segal and dolph ) have thought about putting out the past
few years. van damme is good in the movie, the movie is only worth watching to
van damme fans. it is not as good as wake of death ( which i highly recommend to
anyone of likes van damme ) or in hell but, in my opinion it ' s worth watching.
it has the same type of feel to it as nowhere to run. good fun stuff! [SEP]"]
[tensor(0), "[CLS] i ' m the type of guy who loves hood movies from new jack
city to baby boy to killa season, from the b grade to the hollywood. but this
movie was something different. i am no hater and this movie was kinda enjoyable.
but some bits were just weird. well the acting wasn ' t to good, compared to
silkk the shockers performance in hot boyz ( quite good ) and ice - t in new
```

`jack and svu (great). the scene where corrupt (ice - t) kills the wanna be jamaican dude he says something and lights himself on fire burning both ice - t and the other dude, this kills the jamaican, however ice - t is unharmed, very similar to ice ' s other movie urban menace (which stars both of these actors) were snoops character is supernatural, however after this there is nothing suggested that corrupt is like a demon. when mj (silkk) gets stabbed at first he struggling but after that he fights normally and was stabbed in the thigh - with out blood. and when mj confesses killing a cop cos the cop was beating up his friend benny was weird, benny isn ' t introduced in this movie and the scene isn ' t in the film. it does hold weight to the fact why corrupt wants to kill mj but is still makes u scratch your head. wen jody writes a letter to miss jones character explaining what happened to them afterwords doesn ' t mention what happen 2 other main characters mj and lisa. the film did show the horror and poverty of the ghetto - which plagues the lives of latinos and blacks word wide - was a good part of the film, even though the clip of the projects was re - used thousands of times. and the scene where miles kills the latino brother by crashing his bike at full speed (not wearing a helmet) and running into my latino brothers car would of killed him. the movie was similar to the film urban menace and half the actors were in both of these movies as well as the production team. it was ok tho. but me being from poverty i love hood films, however if u don ' t love em like i do don ' t watch it. only thing saving me from walking out is it reminded me of the first movie i made which was made with 100 dollars, and my love of the genre. < br / > < br / > nathaniel purez [SEP]"]`

The two chosen wrongly classified examples are positive reviews classified as negative: - *"first off let me say, if you haven ' t enjoyed a van damme movie since bloodsport, you probably will not like this movie. most of these movies may not have the best plots or best actors but i enjoy these kinds of movies for what they are. this movie is much better than any of the movies the other action guys (segal and dolph) have thought about putting out the past few years. van damme is good in the movie, the movie is only worth watching to van damme fans. it is not as good as wake of death (which i highly recommend to anyone of likes van damme) or in hell but, in my opinion it ' s worth watching. it has the same type of feel to it as nowhere to run. good fun stuff!" - "i ' m the type of guy who loves hood movies from new jack city to baby boy to killa season, from the b grade to the hollywood. but this movie was something different. i am no hater and this movie was kinda enjoyable. but some bits were just weird. well the acting wasn ' t to good, compared to silkk the shockers performance in hot boyz (quite good) and ice - t in new jack and svu (great). the scene where corrupt (ice - t) kills the wanna be jamaican dude he says something and lights himself on fire burning both ice - t and the other dude, this kills the jamaican, however ice - t is unharmed, very similar to ice ' s other movie urban menace (which stars both of these actors) were snoops character is supernatural, however after this there is nothing suggested that corrupt is like a demon. when mj (silkk) gets stabbed at first he struggling but after that he fights normally and was stabbed in the thigh - with out blood. and when mj confesses killing a cop cos the cop was beating up his friend benny was weird, benny isn ' t introduced in this movie and the scene isn ' t in the film. it does hold weight to the fact why corrupt wants to kill mj but is still makes u scratch your head. wen jody writes a letter to miss jones character explaining what happened to them afterwords doesn ' t mention what happen 2 other main characters mj and lisa. the film did show the horror and poverty of the ghetto - which plagues the lives of latinos and blacks word wide - was a good part of the film, even though the clip of the projects was re - used thousands of times. and the scene where*

miles kills the latino brother by crashing his bike at full speed (not wearing a helmet) and running into my latino brothers car would of killed him. the movie was similar to the film urban menace and half the actors were in both of these movies as well as the production team. it was ok tho. but me being from poverty i love hood films, however if u don ' t love em like i do don ' t watch it. only thing saving me from walking out is it reminded me of the first movie i made which was made with 100 dollars, and my love of the genre."

The first example can be considered as hard to classify, because it does not really explicitly gives a biased opinion. The reviewer explains why not everybody would like the movie, which implies they use negative terms. Only at the end do they really give the positive feedback. So it is pretty understandable that the model is confused with this type of structure, since it does not base the classification purely on the end of the text, but on the whole context (here we are talking about Transformers...).

The second example is in the same vein as the first one, where the reviewer explains that they like this genre of movie but it's likely that not everybody will. Also, this review contains a big description of the movie, which stays around the semantic field of violence and negativity. This could be a part of why the classifier fails to do well. Also, as in the first example, there is no real expression of positivity as a whole, but only some small bits here and there, which does not give the transformer enough power towards this class.

What are the advantages and inconvenient of using this model in production compared to the naive Bayes we implemented in the first part of the course? And compared to a recurrent model like an RNN or an LSTM? The advantages of distilbert model compared to the naive Bayes we implemented is that it takes a lot more information into account. Indeed, the naive Bayes only takes into account the words and their frequency in the sentence. The distilbert model takes into account the context of the words, the order of the words, the meaning of the words, etc. It is a lot more powerful than the naive Bayes model. However, it is also a lot more complex and takes a lot more time to train (naive Bayes is very fast). It is also a lot more difficult to understand and to debug.

The advantages of distilbert model compared to a recurrent model like an RNN or an LSTM is that it is a lot faster to train and to use. Using pre-trained models like distilbert is also a lot easier than training a model from scratch. However, the recurrent models are a lot more flexible and can be used for a lot more tasks than the distilbert model. They can also be used for other languages than English. The recurrent models are also a lot more interpretable than the distilbert model. Indeed, we can see the output of each cell of the RNN and understand what it is doing. It is not the case for the distilbert model.

The model only accepts inputs of maximum 512 tokens. Propose and implement a solution that goes around that. Let's generate a long review using ChatGPT to show our love for Transformers

[42]:

```
long_text = "Eclipse of the Mind is a mind-bending thriller that takes␣
 ↪audiences on a captivating journey through the depths of human consciousness.
 ↪ Directed by visionary filmmaker Adam Jacobs, this psychological␣
 ↪rollercoaster delves into the enigmatic and labyrinthine recesses of the␣
 ↪mind, challenging our perception of reality and the boundaries of our own␣
 ↪sanity. The story revolves around Sarah Thompson (played brilliantly by Emma␣
 ↪Roberts), a talented neuroscientist grappling with the loss of her husband␣
 ↪and a tragic accident that left her with selective amnesia. In her desperate␣
 ↪pursuit to regain her memories, Sarah embarks on a dangerous experimental␣
 ↪procedure that promises to unlock the secrets hidden within her mind. From␣
 ↪the very beginning, the film grabs your attention with its mesmerizing␣
 ↪visuals and a hauntingly atmospheric score that intensifies the eerie␣
 ↪ambiance. As Sarah delves deeper into her subconscious, the line between␣
 ↪dreams and reality becomes blurred, leaving both Sarah and the audience␣
 ↪questioning what is real and what is merely a construct of her fractured␣
 ↪mind. The film's greatest strength lies in its thought-provoking narrative,␣
 ↪which seamlessly weaves together themes of identity, perception, and the␣
 ↪fragility of human memory. The screenplay, penned by Michael Anders, is␣
 ↪filled with intriguing twists and turns that keep you on the edge of your␣
 ↪seat, constantly guessing the true nature of Sarah's reality. The dialogue␣
 ↪is sharp and insightful, delving into the depths of existential questions␣
 ↪while maintaining a sense of urgency and suspense. Emma Roberts delivers a␣
 ↪stellar performance as Sarah, showcasing her versatility as an actress. She␣
 ↪captures the character's vulnerability, determination, and growing paranoia␣
 ↪with precision, drawing the audience into her tumultuous journey. The␣
 ↪supporting cast, including the enigmatic Dr. Jonathan Reed (portrayed by␣
 ↪Michael Shannon) and the enigmatic antagonist played by John Doe, complement␣
 ↪Roberts' performance with their nuanced portrayals, adding layers of␣
 ↪complexity to the unfolding mystery. Visually, Eclipse of the Mind is a␣
 ↪treat for the senses. Jacobs's masterful direction creates a surreal and␣
 ↪haunting world that mirrors the disorienting nature of Sarah's mental state.␣
 ↪The cinematography, with its stark contrasts and clever use of lighting,␣
 ↪amplifies the sense of unease, making each frame visually striking. While␣
 ↪the film is undeniably gripping and intellectually stimulating, it␣
 ↪occasionally falters in pacing, particularly during the second act. Some␣
 ↪scenes could have been trimmed to maintain a more consistent rhythm,␣
 ↪preventing the film from occasionally losing its momentum. Nevertheless,␣
 ↪Eclipse of the Mind is a must-watch for fans of psychological thrillers and␣
 ↪those intrigued by the exploration of the human mind. It is an intelligent␣
 ↪and captivating film that challenges conventional storytelling, leaving you␣
 ↪contemplating the fragility of our own perceptions and the depths of our own␣
 ↪consciousness. With its outstanding performances, intriguing narrative, and␣
 ↪stunning visuals, this mind-bending journey will keep you enthralled from␣
 ↪start to finish."
long_tokens = tokenizer.encode(long_text, add_special_tokens=True)
print(len(long_tokens))
```

```
Token indices sequence length is longer than the specified maximum sequence
length for this model (606 > 512). Running this sequence through the model will
result in indexing errors

606
```

[44]:
```python
# define the sliding window parameters
window_size = 512
stride = 64

# initialize the output tensor
output_tensor = torch.zeros((1, 2)).to(device)

# loop over the input text with a sliding window
for i in range(0, len(long_tokens), stride):
    # get the current window
    window = long_tokens[i : i + window_size]

    # pad the window if necessary
    if len(window) < window_size:
        window += [tokenizer.pad_token_id] * (window_size - len(window))

    # convert the window to a tensor
    window_tensor = torch.tensor(window).unsqueeze(0).to(device)

    # pass the window through the model
    with torch.no_grad():
        logits = model(window_tensor)[0]
        output_tensor += logits

# average the predictions
print(output_tensor)
output_tensor /= len(long_tokens) // stride + 1

# get the predicted class
predicted_class = torch.argmax(output_tensor, dim=1).item()

# get the predicted probability
predicted_probability = torch.softmax(output_tensor, dim=1)[0][predicted_class].
 ↪item()

# print the results
print(f"Predicted class: {predicted_class}")
print(f"Predicted probability: {predicted_probability}")
```

```
tensor([[-26.5703,  31.3720]], device='cuda:0')
Predicted class: 1
Predicted probability: 0.996964156627655
```

The results are the following: - Predicted class: 1 - Predicted probability: 0.996964156627655

We see that even if the tokenizer threw a warning saying the sequence is too long, we managed to predict a class correctly.