Behind the pipeline (PyTorch)

File Edit View Insert Runtime Tools Help




+ Code + Text Copy to Drive

Behind the pipeline (PyTorch)

Install the Transformers, Datasets, and Evaluate libraries to run this notebook.

```
[ ]  !pip install datasets evaluate transformers[sentencepiece]
```

```
[ ]  from transformers import pipeline

     classifier = pipeline("sentiment-analysis")
     classifier(
         [
             "I've been waiting for a HuggingFace course my whole life.",
             "I hate this so much!",
         ]
     )
```

```
[{'label': 'POSITIVE', 'score': 0.9598047137260437},
 {'label': 'NEGATIVE', 'score': 0.9994558095932007}]
```

```
[ ]  from transformers import AutoTokenizer
```

```
checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
tokenizer = AutoTokenizer.from_pretrained(checkpoint)
```

```
[ ] raw_inputs = [
        "I've been waiting for a HuggingFace course my whole life.",
        "I hate this so much!",
    ]
    inputs = tokenizer(raw_inputs, padding=True, truncation=True, return_tensors="pt")
    print(inputs)
```

```
{
    'input_ids': tensor([
        [  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172, 2607,  2026,
        [  101,  1045,  5223,  2023,  2061,  2172,   999,   102,     0,     0,     0,     0,
    ]),
    'attention_mask': tensor([
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]
    ])
}
```

```
[ ] from transformers import AutoModel

    checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
    model = AutoModel.from_pretrained(checkpoint)
```

```
[ ] outputs = model(**inputs)
    print(outputs.last_hidden_state.shape)
```

```
torch.Size([2, 16, 768])
```

```
[ ]  from transformers import AutoModelForSequenceClassification

     checkpoint = "distilbert-base-uncased-finetuned-sst-2-english"
     model = AutoModelForSequenceClassification.from_pretrained(checkpoint)
     outputs = model(**inputs)
```

```
[ ]  print(outputs.logits.shape)

     torch.Size([2, 2])
```

```
[ ]  print(outputs.logits)

     tensor([[-1.5607,  1.6123],
             [ 4.1692, -3.3464]], grad_fn=<AddmmBackward>)
```

```
[ ]  import torch

     predictions = torch.nn.functional.softmax(outputs.logits, dim=-1)
     print(predictions)

     tensor([[4.0195e-02, 9.5980e-01],
             [9.9946e-01, 5.4418e-04]], grad_fn=<SoftmaxBackward>)
```

```
[ ]  model.config.id2label

     {0: 'NEGATIVE', 1: 'POSITIVE'}
```