Processing the data (PyTorch)

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

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

```
[ ] import torch
    from transformers import AdamW, AutoTokenizer, AutoModelForSequenceClassification

    # Same as before
    checkpoint = "bert-base-uncased"
    tokenizer = AutoTokenizer.from_pretrained(checkpoint)
    model = AutoModelForSequenceClassification.from_pretrained(checkpoint)
    sequences = [
        "I've been waiting for a HuggingFace course my whole life.",
        "This course is amazing!",
    ]
    batch = tokenizer(sequences, padding=True, truncation=True, return_tensors="pt")

    # This is new
    batch["labels"] = torch.tensor([1, 1])

    optimizer = AdamW(model.parameters())
    loss = model(**batch).loss
    loss.backward()
    optimizer.step()
```

```
[ ] from datasets import load_dataset

    raw_datasets = load_dataset("glue", "mrpc")
```

```
    validation: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx'],
        num_rows: 408
    })
    test: Dataset({
        features: ['sentence1', 'sentence2', 'label', 'idx'],
        num_rows: 1725
    })
})
```

```
[ ] raw_train_dataset = raw_datasets["train"]
    raw_train_dataset[0]
```

```
{'idx': 0,
 'label': 1,
 'sentence1': 'Amrozi accused his brother , whom he called " the witness " , of deliberately distorting his evidence
 'sentence2': 'Referring to him as only " the witness " , Amrozi accused his brother of deliberately distorting his e
```

```
[ ] raw_train_dataset.features
```

```
{'sentence1': Value(dtype='string', id=None),
 'sentence2': Value(dtype='string', id=None),
 'label': ClassLabel(num_classes=2, names=['not_equivalent', 'equivalent'], names_file=None, id=None),
 'idx': Value(dtype='int32', id=None)}
```

```
[ ] from transformers import AutoTokenizer

    checkpoint = "bert-base-uncased"
    tokenizer = AutoTokenizer.from_pretrained(checkpoint)
    tokenized_sentences_1 = tokenizer(raw_datasets["train"]["sentence1"])
    tokenized_sentences_2 = tokenizer(raw_datasets["train"]["sentence2"])
```

```
[ ] inputs = tokenizer("This is the first sentence.", "This is the second one.")
    inputs
```

```
{
  'input_ids': [101, 2023, 2003, 1996, 2034, 6251, 1012, 102, 2023, 2003, 1996, 2117, 2028, 1012, 102],
```

```
  'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1],
  'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
}
```

```
[ ] tokenizer.convert_ids_to_tokens(inputs["input_ids"])
```

```
['[CLS]', 'this', 'is', 'the', 'first', 'sentence', '.', '[SEP]', 'this', 'is', 'the', 'second', 'one', '.', '[SEP]']
```

```
[ ] tokenized_dataset = tokenizer(
        raw_datasets["train"]["sentence1"],
        raw_datasets["train"]["sentence2"],
        padding=True,
        truncation=True,
    )
```

```
[ ] def tokenize_function(example):
        return tokenizer(example["sentence1"], example["sentence2"], truncation=True)
```

```
[ ] tokenized_datasets = raw_datasets.map(tokenize_function, batched=True)
    tokenized_datasets
```

```
DatasetDict({
    train: Dataset({
        features: ['attention_mask', 'idx', 'input_ids', 'label', 'sentence1', 'sentence2', 'token_type_ids'],
        num_rows: 3668
    })
    validation: Dataset({
        features: ['attention_mask', 'idx', 'input_ids', 'label', 'sentence1', 'sentence2', 'token_type_ids'],
        num_rows: 408
    })
    test: Dataset({
        features: ['attention_mask', 'idx', 'input_ids', 'label', 'sentence1', 'sentence2', 'token_type_ids'],
        num_rows: 1725
    })
})
```

```
[ ] from transformers import DataCollatorWithPadding
```

```
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

```
[ ] samples = tokenized_datasets["train"][:8]
    samples = {k: v for k, v in samples.items() if k not in ["idx", "sentence1", "sentence2"]}
    [len(x) for x in samples["input_ids"]]
```

```
[50, 59, 47, 67, 59, 50, 62, 32]
```

```
[ ] batch = data_collator(samples)
    {k: v.shape for k, v in batch.items()}
```

```
{'attention_mask': torch.Size([8, 67]),
 'input_ids': torch.Size([8, 67]),
 'token_type_ids': torch.Size([8, 67]),
 'labels': torch.Size([8])}
```