Handling multiple sequences (PyTorch)

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

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

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

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

    sequence = "I've been waiting for a HuggingFace course my whole life."

    tokens = tokenizer.tokenize(sequence)
    ids = tokenizer.convert_tokens_to_ids(tokens)
    input_ids = torch.tensor(ids)
    # This line will fail.
    model(input_ids)
```

```
IndexError: Dimension out of range (expected to be in range of [-1, 0], but got 1)
```

```
[ ] tokenized_inputs = tokenizer(sequence, return_tensors="pt")
    print(tokenized_inputs["input_ids"])
```

```
tensor([[  101,  1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,
          2607,  2026,  2878,  2166,  1012,   102]])
```

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

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

     sequence = "I've been waiting for a HuggingFace course my whole life."

     tokens = tokenizer.tokenize(sequence)
     ids = tokenizer.convert_tokens_to_ids(tokens)

     input_ids = torch.tensor([ids])
     print("Input IDs:", input_ids)

     output = model(input_ids)
     print("Logits:", output.logits)
```

```
Input IDs: [[ 1045,  1005,  2310,  2042,  3403,  2005,  1037, 17662, 12172,  2607, 2026,  2878,  2166,  101
Logits: [[-2.7276,  2.8789]]
```

```
[ ]  batched_ids = [
         [200, 200, 200],
         [200, 200]
     ]
```

```
[ ]  padding_id = 100

     batched_ids = [
         [200, 200, 200],
         [200, 200, padding_id],
```

```
]
```

```
[ ] model = AutoModelForSequenceClassification.from_pretrained(checkpoint)

    sequence1_ids = [[200, 200, 200]]
    sequence2_ids = [[200, 200]]
    batched_ids = [
        [200, 200, 200],
        [200, 200, tokenizer.pad_token_id],
    ]

    print(model(torch.tensor(sequence1_ids)).logits)
    print(model(torch.tensor(sequence2_ids)).logits)
    print(model(torch.tensor(batched_ids)).logits)
```

```
tensor([[ 1.5694, -1.3895]], grad_fn=<AddmmBackward>)
tensor([[ 0.5803, -0.4125]], grad_fn=<AddmmBackward>)
tensor([[ 1.5694, -1.3895],
        [ 1.3373, -1.2163]], grad_fn=<AddmmBackward>)
```

```
[ ] batched_ids = [
        [200, 200, 200],
        [200, 200, tokenizer.pad_token_id],
    ]

    attention_mask = [
        [1, 1, 1],
        [1, 1, 0],
    ]

    outputs = model(torch.tensor(batched_ids), attention_mask=torch.tensor(attention_mask))
    print(outputs.logits)
```

```
tensor([[ 1.5694, -1.3895],
        [ 0.5803, -0.4125]], grad_fn=<AddmmBackward>)
```

```
[ ]  sequence = sequence[:max_sequence_length]
```