20240328 Meeting

312707003 黃鈺婷

預處理數據

- 將文本轉換為模型能夠理解的數字

```
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"])
```

預處理數據

- 也可以直接輸入一組句子

```
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]']
```

動態填充

```
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])}
```

- 將該批中的所有樣本都填充到最大長度
- 若沒有動態填充，所有的樣本都須填充到整個數據集中的最大長度

Training

- 定義TrainingArguments，它包含Trainer用於訓練和評估的所有超參數

```
from transformers import TrainingArguments

training_args = TrainingArguments("test-trainer")
```

- 定義模型

```
from  transformers  import  AutoModelForSequenceClassification

model  =  AutoModelForSequenceClassification.from_pretrained(checkpoint,  num_labels=2)
```

Training

- 定義Trainer

```
from transformers import Trainer

trainer = Trainer(
    model,
    training_args,
    train_dataset=tokenized_datasets["train"],
    eval_dataset=tokenized_datasets["validation"],
    data_collator=data_collator,
    tokenizer=tokenizer,
)
```

Training

- 讓預訓練模型在數據集上微調

```
trainer.train()
```

- 沒有提供一個compute_metrics()函數來直接計算模型的好壞，所以只會每500步報告一次訓練損失

compute_metrics()

```
import numpy as np

preds = np.argmax(predictions.predictions, axis=-1)
```

```
import evaluate

metric = evaluate.load("glue", "mrpc")
metric.compute(predictions=preds, references=predictions.label_ids)
```

```
{'accuracy': 0.8578431372549019, 'f1': 0.8996539792387542}
```

可以直接看出模型好壞

- 定義compute_metrics()函數

```
def compute_metrics(eval_preds):
    metric = evaluate.load("glue", "mrpc")
    logits, labels = eval_preds
    predictions = np.argmax(logits, axis=-1)
    return metric.compute(predictions=predictions, references=labels)
```

新的Trainer

```
training_args = TrainingArguments("test-trainer", evaluation_strategy="epoch")
model = AutoModelForSequenceClassification.from_pretrained(checkpoint, num_labels=2)

trainer = Trainer(
        model,
        training_args,
        train_dataset=tokenized_datasets["train"],
        eval_dataset=tokenized_datasets["validation"],
        data_collator=data_collator,
        tokenizer=tokenizer,
        compute_metrics=compute_metrics,
)
```

創建驗證集

- 使用Dataset.train_test_split()將訓練集分成train和validation

```
drug_dataset_clean = drug_dataset["train"].train_test_split(train_size=0.8, seed=42)
# Rename the default "test" split to "validation"
drug_dataset_clean["validation"] = drug_dataset_clean.pop("test")
# Add the "test" set to our `DatasetDict`
drug_dataset_clean["test"] = drug_dataset["test"]
drug_dataset_clean
```

```
DatasetDict({
    train: Dataset({
        features: ['patient_id', 'drugName', 'condition', 'review', 'rating', 'date', 'usefulCount',
'review_length', 'review_clean'],
        num_rows: 110811
    })
    validation: Dataset({
        features: ['patient_id', 'drugName', 'condition', 'review', 'rating', 'date', 'usefulCount',
'review_length', 'review_clean'],
        num_rows: 27703
    })
    test: Dataset({
        features: ['patient_id', 'drugName', 'condition', 'review', 'rating', 'date', 'usefulCount',
'review_length', 'review_clean'],
        num_rows: 46108
    })
})
```

Streaming dataset

```
pubmed_dataset_streamed = load_dataset(
    "json", data_files=data_files, split="train", streaming=True
)
```

會產生 IterableDataset

- Streaming dataset第一個元素

```
next(iter(pubmed_dataset_streamed))
```

使用FAISS進行相似性搜索

- 先使用Dataset.add_faiss_index()函數指定要索引數據集的哪一列

```
embeddings_dataset.add_faiss_index(column="embeddings")
```

- 再使用Dataset.get_nearest_examples()函數返回一個分數元組，對查詢和文檔之間的相似度進行排序，找最佳匹配結果

```
scores, samples = embeddings_dataset.get_nearest_examples(
    "embeddings", question_embedding, k=5
)
```