Tiền xử lý

Ngoài việc tải tập dữ liệu, Mục tiêu chính khác của bộ dữ liệu là cung cấp một tập hợp đa dạng các chức năng tiền xử lý để chuyển tập dữ liệu sang định dạng thích hợp cho việc đào tạo với khung học máy.

Có nhiều cách có thể để xử lý trước một tập dữ liệu và tất cả đều phụ thuộc vào yêu cầu cụ thể của bạn. tập dữ liệu. Đôi khi bạn có thể cần đổi tên một cột và những lúc khác bạn có thể cần phải đổi tên các trường lồng nhau không phẳng. Bộ dữ liệu cung cấp cách để thực hiện hầu hết những việc này. Nhưng t các trường hợp tiền xử lý, tùy thuộc vào phương thức tập dữ liệu của bạn, bạn sẽ cần:

- Mã hóa một tập dữ liệu văn bản.
- Lấy mẫu lại tập dữ liệu âm thanh.
- Áp dụng các phép biến đổi cho tập dữ liệu hình ảnh.

Bước tiền xử lý cuối cùng thường là đặt định dạng tập dữ liệu của bạn để tương thích với định dạng đầu vào dự kiến của khung học máy.

Trong hướng dẫn này, bạn cũng cần cài đặt thư viện Transformers:

```
pip cài đặt máy biến áp
```

Lấy một tập dữ liệu bạn chọn và làm theo!

Mã hóa văn bản

Mô hình không thể xử lý văn bản thô, vì vậy bạn sẽ cần chuyển đổi văn bản thành số. Mã thông báo cung cấp cách thực hiện việc này bằng cách chia văn bản thành các từ riêng lẻ được gọi là mã thông báo. Mã được chuyển thành số.

> [!TIP]
> Hãy xem phần Tokenizers trong Chương 2 của khóa học Ôm mặt để tìm hiểu thêm về mã thông báo và các thuật toán mã thông báo khác nhau.

1. Bắt đầu bằng cách tải tập dữ liệu rotten_tomatoes và mã thông báo tương ứng với mô hình BERT được huấn luyện trước. Việc sử dụng cùng một mã thông báo như mô hình được đào tạo trướ

bởi vì bạn muốn đảm bảo văn bản được phân chia theo cùng một cách.

```
>>> from transformers import AutoTokenizer
>>> from datasets import load_dataset

>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-uncased")
>>> dataset = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
```

2. Gọi mã thông báo của bạn trên hàng văn bản đầu tiên trong tập dữ liệu:

```
>>> tokenizer(dataset[0]["text"])
{'input_ids': [101, 1103, 2067, 1110, 17348, 1106, 1129, 1103, 6880, 1432, 112, 188, 1207, 107, 14
 'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 'attention_mask': [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

Trình mã thông báo trả về một từ điển có ba mục:

- input_ids : các số đại diện cho token trong văn bản.
- token_type_ids : cho biết mã thông báo thuộc về chuỗi nào nếu có nhiều hơn một mã thông báo sự liên tiếp.
- Chú ý_mask : cho biết liệu mã thông báo có nên được che giấu hay không.

Những giá trị này thực sự là đầu vào của mô hình.

3. The fastest way to tokenize your entire dataset is to use the map() function. This function tăng tốc độ mã thông báo bằng cách áp dụng mã thông báo cho các lô ví dụ thay vì riêng lẻ ví dụ. Đặt tham số theo đợt thành True :

```
>>> def tokenization(example):
...return tokenizer(example["text"])

>>> dataset = dataset.map(tokenization, batched=True)
```

4. Đặt định dạng của tập dữ liệu để tương thích với khung học máy của bạn:

Use the [set_format()](/docs/datasets/v4.2.0/en/package_reference/main_classes#datasets.Dataset.set_format) function to set the dataset format to be compatible

với PyTorch:

```
>>> dataset.set_format(type="torch", columns=["input_ids", "token_type_ids", "attention_mask", "la
>>> dataset.format['type']
'ngọn đuốc'
```

Use the [to_tf_dataset()](/docs/datasets/v4.2.0/en/package_reference/main_classes#datasets.Dataset.to_tf_dataset) function to set the dataset format to be compatible with TensorFlow. You'll also need to import a [data collator](https://huggingface.co/docs/transformers/main_classes/data_collator#transformers.DataCollatorWithPadding) from
Máy biến áp để kết hợp các độ dài chuỗi khác nhau thành một lô có độ dài bằng nhau:

```
>>> from transformers import DataCollatorWithPadding

>>> data_collator = DataCollatorWithPadding(tokenizer=tokenizer, return_tensors="tf")
>>> tf_dataset = dataset.to_tf_dataset(
...columns=["input_ids", "token_type_ids", "attention_mask"],
...label_cols=["label"],
...batch_size=2,
...collate_fn=data_collator,
...shuffle=True
... )
```

5. Tập dữ liệu hiện đã sẵn sàng để đào tạo với khung học máy của bạn!

Lấy mẫu lại tín hiệu âm thanh

Đầu vào âm thanh như tập dữ liệu văn bản cần được chia thành các điểm dữ liệu riêng biệt. Điều này được g
sampling; the sampling rate tells you how much of the speech signal is captured per second. It
điều quan trọng là đảm bảo tốc độ lấy mẫu của tập dữ liệu của bạn khớp với tốc độ lấy mẫu của
dữ liệu được sử dụng để huấn luyện trước mô hình bạn đang sử dụng. Nếu tốc độ lấy mẫu khác nhau, dữ liệ
mô hình có thể hoạt động kém trên tập dữ liệu của bạn vì nó không nhận ra sự khác biệt trong
tốc độ lấy mẫu

1. Bắt đầu bằng cách tải tập dữ liệu MInDS-14, tính năng Âm thanh và trình trích xuất tính năng
tương ứng với mô hình Wav2Vec2 đã được huấn luyện trước:

```
>>> from transformers import AutoFeatureExtractor
>>> from datasets import load_dataset, Audio

>>> feature_extractor = AutoFeatureExtractor.from_pretrained("facebook/wav2vec2-base-960h")
>>> dataset = load_dataset("PolyAI/minds14", "en-US", split="train")
```

2. Lập chỉ mục vào hàng đầu tiên của tập dữ liệu. Khi bạn gọi cột âm thanh của tập dữ liệu, nó là tự động giải mã và lấy mẫu lại:

```
>>> audio = dataset[0]["audio"]
>>> print(audio)
<datasets.features._torchcodec.AudioDecoder object at 0x11642b6a0>
>>> audio.get_all_samples().sample_rate
8000
```

3. Đọc thẻ tập dữ liệu cực kỳ hữu ích và có thể cung cấp cho bạn nhiều thông tin về tập dữ liệu. Nhìn nhanh vào thẻ dữ liệu MInDS-14 sẽ cho bạn biết tốc độ lấy mẫu là 8kHz. Tương tự như vậy, bạn có thể nhận được nhiều thông tin chi tiết về một mô hình từ thẻ mô hình của nó. Mô thẻ cho biết nó đã được lấy mẫu trên âm thanh giọng nói 16kHz. Điều này có nghĩa là bạn sẽ cần lấy mẫu lại Tập dữ liệu MInDS-14 để phù hợp với tốc độ lấy mẫu của mô hình.

Use the cast_column() function and set the sampling_rate parameter in the Audio feature to lấy mẫu tín hiệu âm thanh. Khi bạn gọi cột âm thanh bây giờ, nó sẽ được giải mã và lấy mẫu lại đến 16kHz:

```
>>> dataset = dataset.cast_column("audio", Audio(sampling_rate=16_000))
>>> audio = dataset[0]["audio"]
>>> print(audio)
<datasets.features._torchcodec.AudioDecoder object at 0x11642b6a0>
>>> audio.get_all_samples().sample_rate
16000
```

4. Use the map() function to resample the entire dataset to 16kHz. This function speeds up lấy mẫu lại bằng cách áp dụng trình trích xuất tính năng cho hàng loạt mẫu thay vì riêng lẻ ví dụ. Đặt tham số theo đợt thành True :

```
>>> def preprocess_function(examples):
...audio_arrays = [x.get_all_samples().data for x in examples["audio"]]
...inputs = feature_extractor(
...audio_arrays, sampling_rate=feature_extractor.sampling_rate, max_length=16000, truncat
...)
...trả lại đầu vào

>>> dataset = dataset.map(preprocess_function, batched=True)
```

5. Tập dữ liệu hiện đã sẵn sàng để đào tạo với khung học máy của bạn!

Áp dụng tăng cường dữ liệu

Quá trình tiền xử lý phổ biến nhất mà bạn sẽ thực hiện với tập dữ liệu hình ảnh là tăng cường dữ liệu, một quá trình đưa ra các biến thể ngẫu nhiên cho một hình ảnh mà không làm thay đổi ý nghĩa của dữ liệu. Điều này có thể có nghĩa là thay đổi thuộc tính màu sắc của hình ảnh hoặc cắt xén ngẫu nhiên một h hình ảnh. Bạn có thể tự do sử dụng bất kỳ thư viện tăng cường dữ liệu nào bạn thích và Bộ dữ liệu sẽ trợ giú bạn áp dụng các phần bổ sung dữ liệu cho tập dữ liệu của mình.

1. Bắt đầu bằng cách tải tập dữ liệu Beans, tính năng Hình ảnh và trình trích xuất tính năng tương ứng với mô hình ViT được huấn luyện trước:

```
>>> from transformers import AutoFeatureExtractor
>>> from datasets import load_dataset, Image

>>> feature_extractor = AutoFeatureExtractor.from_pretrained("google/vit-base-patch16-224-in21k")
>>> dataset = load_dataset("AI-Lab-Makerere/beans", split="train")
```

2. Lập chỉ mục vào hàng đầu tiên của tập dữ liệu. Khi bạn gọi cột hình ảnh của tập dữ liệu, đối tượng PIL cơ bản được tự động giải mã thành hình ảnh.

```
>>> dataset[0]["image"]
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x7FE5A047CC70>
```

Hầu hết các mẫu hình ảnh đều mong muốn hình ảnh ở chế độ RGB. Hình ảnh Beans đã có rồi ở chế độ RGB, nhưng nếu tập dữ liệu của bạn chứa hình ảnh ở chế độ khác, bạn có thể sử dụng

cast_column() function to set the mode to RGB:

```
>>> dataset = dataset.cast_column("image", Image(mode="RGB"))
```

3. Bây giờ hãy áp dụng tăng cường dữ liệu cho hình ảnh của bạn. Bộ dữ liệu hoạt động với bất kỳ thư viện tăng cường và trong ví dụ này, chúng tôi sẽ sử dụng Albumentations.

Albumentations là một thư viện tăng cường hình ảnh phổ biến cung cấp một bộ biến đổi phong phú bao gồm các phép biến đổi cấp độ không gian, các phép biến đổi cấp pixel và các phép biến đổi cấp độ trộn.

Cài đặt Albumentations:

```
cài đặt pip
```

4. Tạo một quy trình tăng cường điển hình với Albumentations:

```
>>> import albumentations as A
>>> import numpy as np
>>> from PIL import Image

>>> transform = A.Compose([
...A.RandomCrop(height=256, width=256, pad_if_needed=True, p=1),
...A.HorizontalFlip(p=0.5),
...A.ColorJitter(p=0.5)
... ])
```

5. Vì Bộ dữ liệu sử dụng hình ảnh PIL nhưng Albumentations yêu cầu mảng NumPy, bạn cần phải chuyển đổi giữa các định dạng:

```
>>> def albumentations_transforms(examples):
...# Áp dụng các phép biến đổi Albumentations
...transformed_images = []
...for image in examples["image"]:
...# Convert PIL to numpy array (OpenCV format)
...image_np = np.array(image.convert("RGB"))
None
...# Áp dụng các phép biến đổi Albumentations
...transformed_image = transform(image=image_np)["image"]
None
...# Chuyển về ảnh PIL
...pil_image = Image.fromarray(transformed_image)
...transformed_images.append(pil_image)
None
...examples["pixel_values"] = transformed_images
...trả lại ví dụ
```

6. Apply the transform using with_transform():

```
>>> dataset = dataset.with_transform(albumentations_transforms)
>>> dataset[0]["pixel_values"]
```

Những điểm chính khi sử dụng Albumentations với Bộ dữ liệu:

- Chuyển đổi hình ảnh PIL sang mảng NumPy trước khi áp dụng các phép biến đổi
- Albumentations trả về một từ điển có hình ảnh được chuyển đổi dưới khóa "hình ảnh"
- Chuyển đổi kết quả về định dạng PIL sau khi chuyển đổi

7. Tập dữ liệu hiện đã sẵn sàng để đào tạo với khung học máy của bạn!