Sử dụng với PyArrow

Tài liệu này là phần giới thiệu nhanh về cách sử dụng bộ dữ liệu với PyArrow, với trọng tâm cụ thể là về cách xử lý
tập dữ liệu sử dụng hàm tính toán Arrow và cách chuyển đổi tập dữ liệu sang PyArrow hoặc từ PyArrow.

Điều này đặc biệt hữu ích vì nó cho phép thực hiện các thao tác không sao chép nhanh chóng vì bộ dữ liệu su
dưới mui xe.

Định dạng tập dữ liệu

Theo mặc định, bộ dữ liệu trả về các đối tượng Python thông thường: số nguyên, số float, chuỗi, danh sách, v

Thay vào đó, để nhận Bảng hoặc Mảng PyArrow, bạn có thể đặt định dạng của tập dữ liệu thành pyarrow using Dataset.with_format():

```
>>> from datasets import Dataset
>>> data = {"col_0": ["a", "b", "c", "d"], "col_1": [0., 0., 1., 1.]}
>>> ds = Dataset.from_dict(data)
>>> ds = ds.with_format("arrow")
>>> ds[0]# pa.Table
pyarrow.Bảng
col_0: chuỗi
col_1: gấp đôi
None
col_0: [["a"]]
col_1: [[0]]
>>> ds[:2]# pa.Table
pyarrow.Bảng
col_0: chuỗi
col_1: gấp đôi
None
col_0: [["a","b"]]
col_1: [[0,0]]
>>> ds["data"]# pa.array
<pyarrow.lib.ChunkedArray object at 0x1394312a0>
[
  [
    "Một",
    "b",
    "c",
    "d"
  ]
]
```

Điều này cũng hoạt động đối với các đối tượng IterableDataset thu được, ví dụ: sử dụng load_dataset(..., streaming=True) :

```
>>> ds = ds.with_format("arrow")
>>> for table in ds.iter(batch_size=2):
...print(table)
...phá vỡ
pyarrow.Bảng
col_0: chuỗi
col_1: gấp đôi
None
col_0: [["a","b"]]
col_1: [[0,0]]
```

Xử lý dữ liệu

Các hàm PyArrow thường nhanh hơn các hàm python viết tay thông thường và
do đó chúng là một lựa chọn tốt để tối ưu hóa việc xử lý dữ liệu. Bạn có thể sử dụng tính toán Mũi tên functions to process a dataset in Dataset.map() or Dataset.filter():

```
>>> import pyarrow.compute as pc
>>> from datasets import Dataset
>>> data = {"col_0": ["a", "b", "c", "d"], "col_1": [0., 0., 1., 1.]}
>>> ds = Dataset.from_dict(data)
>>> ds = ds.with_format("arrow")
>>> ds = ds.map(lambda t: t.append_column("col_2", pc.add(t["col_1"], 1)), batched=True)
>>> ds[:2]
pyarrow.Bảng
col_0: chuỗi
col_1: gấp đôi
col_2: gấp đôi
None
col_0: [["a","b"]]
col_1: [[0,0]]
col_2: [[1,1]]
>>> ds = ds.filter(lambda t: pc.equal(t["col_0"], "b"), batched=True)
>>> ds[0]
pyarrow.Bảng
col_0: chuỗi
col_1: gấp đôi
col_2: gấp đôi
None
col_0: [["b"]]
col_1: [[0]]
col_2: [[1]]
```

We use batched=True because it is faster to process batches of data in PyArrow rather than row by row. It's also possible to use batch_size= in map() to set the size of each table .

This also works for IterableDataset.map() and IterableDataset.filter().

Nhập hoặc xuất từ PyArrow

Tập dữ liệu là một trình bao bọc của Bảng PyArrow, bạn có thể khởi tạo Tập dữ liệu trực tiếp từ Bàn:

```
ds = Dataset(table)
```

Bạn có thể truy cập Bảng PyArrow của tập dữ liệu bằng cách sử dụng Dataset.data, nó trả về một MemoryMappedTable hoặc InMemoryTable hoặc ConcatenationTable , tùy thuộc vào nguồn gốc của dữ liệu Mũi tên và các thao tác đã được áp dụng.

Các đối tượng đó bao bọc bảng PyArrow bên dưới có thể truy cập được tại Dataset.data.table . bảng này chứa tất cả dữ liệu của tập dữ liệu, nhưng cũng có thể có ánh xạ chỉ mục tại Dataset._indices ánh xạ các chỉ mục hàng của tập dữ liệu tới các chỉ mục hàng của Bảng PyArrow. This can happen if the dataset has been shuffled with Dataset.shuffle() or if only a subset of the rows are used (e.g. after a Dataset.select()).

Trong trường hợp chung, bạn có thể xuất tập dữ liệu sang Bảng PyArrow bằng cách sử dụng table = ds.with_format("arrow")[:] .