Sử dụng với Polar

Tài liệu này là phần giới thiệu nhanh về cách sử dụng bộ dữ liệu với Polars, đặc biệt tập trung vào cách xử lý
tập dữ liệu sử dụng hàm Polars và cách chuyển đổi tập dữ liệu sang Polars hoặc từ Polars.

Điều này đặc biệt hữu ích vì nó cho phép thực hiện các thao tác không sao chép nhanh chóng, vì cả tập dữ liệ
Polars sử dụng Arrow 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 có được Polars DataFrames hoặc Series, bạn có thể đặt định dạng của tập dữ liệu thành các
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("polars")
>>> ds[0]# pl.DataFrame
shape: (1, 2)
None
 col_0  col_1
None
str f64
None
 một 0,0
None
>>> ds[:2]# pl.DataFrame
shape: (2, 2)
None
 col_0  col_1
None
str f64
None
 một 0,0
 b 0,0
None
>>> ds["data"]# pl.Series
shape: (4,)
Series: 'col_0' [str]
[
        "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("polars")
>>> for df in ds.iter(batch_size=2):
...print(df)
...phá vỡ
shape: (2, 2)
None
 col_0  col_1
None
str f64
None
 một 0,0
 b 0,0
None
```

Xử lý dữ liệu

Các hàm cực 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 các hàm Polars để xử lý một dataset in Dataset.map() or Dataset.filter():

```
>>> import polars as pl
>>> 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("polars")
>>> ds = ds.map(lambda df: df.with_columns(pl.col("col_1").add(1).alias("col_2")), batched=True)
>>> ds[:2]
shape: (2, 3)
None
 col_0  col_1  col_2
None
str f64  f64
None
 một 0,01,0
 b 0,01,0
None
>>> ds = ds.filter(lambda df: df["col_0"] == "b", batched=True)
>>> ds[0]
shape: (1, 3)
None
 col_0  col_1  col_2
None
str f64  f64
None
 b 0,01,0
None
```

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

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

Ví dụ: trích xuất dữ liệu

Nhiều hàm có sẵn trong Polars và cho bất kỳ kiểu dữ liệu nào: chuỗi, số float, số nguyên, v.v. Bạn có thể tìm thấy danh sách đầy đủ ở đây. Các chức năng đó được viết bằng Rust và chạy trên các lô dữ liệu giúp xử lý dữ liệu nhanh chóng.

Dưới đây là ví dụ cho thấy tốc độ tăng gấp 5 lần khi sử dụng Polars thay vì trăn thông thường

chức năng trích xuất các giải pháp từ bộ dữ liệu lý luận LLM:

```
từ tập dữ liệu nhập tải_dataset

ds = load_dataset("ServiceNow-AI/R1-Distill-SFT", "v0", split="train")

# Sử dụng hàm python thông thường
pattern = re.compile("boxed\\{(.*)\\}")
result_ds = ds.map(lambda x: {"value_solution": m.group(1) if (m:=pattern.search(x["solution"])) e
# Thời gian: 10s

# Sử dụng chức năng Polars
expr = pl.col("solution").str.extract("boxed\\{(.*)\\}").alias("value_solution")
result_ds = ds.with_format("polars").map(lambda df: df.with_columns(expr), batched=True)
# Thời gian: 2s
```

Nhập hoặc xuất từ Polars

To import data from Polars, you can use Dataset.from_polars() :

```
ds = Dataset.from_polars(df)
```

And you can use Dataset.to_polars() to export a Dataset to a Polars DataFrame:

```
df = Dataset.to_polars(ds)
```