Use with Spark

This document is a quick introduction to using Datasets with Spark, with a particular focus on how to load a Spark DataFrame into a Dataset object.

From there, you have fast access to any element and you can use it as a data loader to train models.

Load from Spark

A Dataset object is a wrapper of an Arrow table, which allows fast reads from arrays in the dataset to PyTorch, TensorFlow and JAX tensors.
The Arrow table is memory mapped from disk, which can load datasets bigger than your available RAM.

You can get a Dataset from a Spark DataFrame using `Dataset.from_spark()`:

```
>>> from datasets import Dataset
>>> df = spark.createDataFrame(
...     data=[[1, "Elia"], [2, "Teo"], [3, "Fang"]],
...     columns=["id", "name"],
... )
>>> ds = Dataset.from_spark(df)
```

The Spark workers write the dataset on disk in a cache directory as Arrow files, and the Dataset is loaded from there.

Alternatively, you can skip materialization by using `IterableDataset.from_spark()`, which returns an IterableDataset:

```
>>> from datasets import IterableDataset
>>> df = spark.createDataFrame(
...     data=[[1, "Elia"], [2, "Teo"], [3, "Fang"]],
...     columns=["id", "name"],
... )
>>> ds = IterableDataset.from_spark(df)
>>> print(next(iter(ds)))
{"id": 1, "name": "Elia"}
```

Caching

When using `Dataset.from_spark()`, the resulting Dataset is cached; if you call `Dataset.from_spark()` multiple
times on the same DataFrame it won't re-run the Spark job that writes the dataset as Arrow files on disk.

You can set the cache location by passing `cache_dir=` to `Dataset.from_spark()`.
Make sure to use a disk that is available to both your workers and your current machine (the driver).

> [!WARNING]
> In a different session, a Spark DataFrame doesn't have the same semantic hash, and it will rerun a Spark job and store it in a new cache.

Feature types

If your dataset is made of images, audio data or N-dimensional arrays, you can specify the `features=` argument in
`Dataset.from_spark()` (or `IterableDataset.from_spark()`):

```
>>> from datasets import Dataset, Features, Image, Value
>>> data = [(0, open("image.png", "rb").read())]
>>> df = spark.createDataFrame(data, "idx: int, image: binary")
>>> # Also works if you have arrays
>>> # data = [(0, np.zeros(shape=(32, 32, 3), dtype=np.int32).tolist())]
>>> # df = spark.createDataFrame(data, "idx: int, image: array<array<array<int>>>")
>>> features = Features({"idx": Value("int64"), "image": Image()})
>>> dataset = Dataset.from_spark(df, features=features)
>>> dataset[0]
{'idx': 0, 'image': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=32x32>}
```

You can check the Features documentation to know about all the feature types available.