Know your dataset

There are two types of dataset objects, a regular Dataset and then an ✨ IterableDataset ✨. A Dataset provides fast random access to the rows, and memory-mapping so that loading even large datasets only uses a relatively small amount of device memory. But for really, really big datasets that won't even fit on disk or in memory, an IterableDataset allows you to access and use the dataset without waiting for it to download completely!

This tutorial will show you how to load and access a Dataset and an IterableDataset.

Dataset

When you load a dataset split, you'll get a Dataset object. You can do many things with a Dataset object, which is why it's important to learn how to manipulate and interact with the data stored inside.

This tutorial uses the rotten_tomatoes dataset, but feel free to load any dataset you'd like and follow along!

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
```

Indexing

A Dataset contains columns of data, and each column can be a different type of data. The *index*, or axis label, is used to access examples from the dataset. For example, indexing by the row returns a dictionary of an example from the dataset:

```
# Get the first row in the dataset
>>> dataset[0]
{'label': 1,
 'text': 'the rock is destined to be the 21st century\'s new " conan " and that he\'s going to mak
```

Use the `-` operator to start from the end of the dataset:

```
# Get the last row in the dataset
>>> dataset[-1]
{'label': 0,
 'text': 'things really get weird , though not particularly scary : the movie is all portent and n
```

Indexing by the column name returns a list of all the values in the column:

```
>>> dataset["text"]
['the rock is destined to be the 21st century\'s new " conan " and that he\'s going to make a spla
 'the gorgeously elaborate continuation of " the lord of the rings " trilogy is so huge that a col
 'effective but too-tepid biopic',
 ...,
 'things really get weird , though not particularly scary : the movie is all portent and no conten
```

You can combine row and column name indexing to return a specific value at a position:

```
>>> dataset[0]["text"]
'the rock is destined to be the 21st century\'s new " conan " and that he\'s going to make a splas
```

Indexing order doesn't matter. Indexing by the column name first returns a Column object that you can index as usual with row indices:

```
>>> import time

>>> start_time = time.time()
>>> text = dataset[0]["text"]
>>> end_time = time.time()
>>> print(f"Elapsed time: {end_time - start_time:.4f} seconds")
Elapsed time: 0.0031 seconds

>>> start_time = time.time()
>>> text = dataset["text"][0]
>>> end_time = time.time()
>>> print(f"Elapsed time: {end_time - start_time:.4f} seconds")
Elapsed time: 0.0042 seconds
```

Slicing

Slicing returns a slice - or subset - of the dataset, which is useful for viewing several rows at once. To slice a dataset, use the `:` operator to specify a range of positions.

```
# Get the first three rows
>>> dataset[:3]
{'label': [1, 1, 1],
 'text': ['the rock is destined to be the 21st century\'s new " conan " and that he\'s going to ma
  'the gorgeously elaborate continuation of " the lord of the rings " trilogy is so huge that a co
  'effective but too-tepid biopic']}

# Get rows between three and six
>>> dataset[3:6]
{'label': [1, 1, 1],
 'text': ['if you sometimes like to go to the movies to have fun , wasabi is a good place to start
  "emerges as something rare , an issue movie that's so honest and keenly observed that it doesn't
  'the film provides some great insight into the neurotic mindset of all comics -- even those who
```

IterableDataset

An IterableDataset is loaded when you set the `streaming` parameter to `True` in load_dataset():

```
>>> from datasets import load_dataset

>>> iterable_dataset = load_dataset("ethz/food101", split="train", streaming=True)
>>> for example in iterable_dataset:
...     print(example)
...     break
{'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=384x512 at 0x7F0681F5C520>, 'labe
```

You can also create an IterableDataset from an *existing* Dataset, but it is faster than streaming mode because the dataset is streamed from local files:

```
>>> from datasets import load_dataset

>>> dataset = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> iterable_dataset = dataset.to_iterable_dataset()
```

An IterableDataset progressively iterates over a dataset one example at a time, so you don't have to wait for the whole dataset to download before you can use it. As you can imagine, this is quite useful for large datasets you want to use immediately!

Indexing

An IterableDataset's behavior is different from a regular Dataset. You don't get random access to examples in an IterableDataset. Instead, you should iterate over its elements, for example, by calling `next(iter())` or with a `for` loop to return the next item from the IterableDataset:

```
>>> next(iter(iterable_dataset))
{'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=384x512 at 0x7F0681F59B50>,
 'label': 6}

>>> for example in iterable_dataset:
...     print(example)
...     break
{'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=384x512 at 0x7F7479DE82B0>, 'labe
```

But an IterableDataset supports column indexing that returns an iterable for the column values:

```
>>> next(iter(iterable_dataset["label"]))
6
```

Creating a subset

You can return a subset of the dataset with a specific number of examples in it with IterableDataset.take():

```
# Get first three examples
>>> list(iterable_dataset.take(3))
[{'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=384x512 at 0x7F7479DEE9D0>,
  'label': 6},
 {'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=512x512 at 0x7F7479DE8190>,
  'label': 6},
 {'image': <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=512x383 at 0x7F7479DE8310>,
  'label': 6}]
```

But unlike slicing, IterableDataset.take() creates a new IterableDataset.

Next steps

Interested in learning more about the differences between these two types of datasets? Learn more about them in the Differences between `Dataset` and `IterableDataset` conceptual guide.

To get more hands-on with these datasets types, check out the Process guide to learn how to preprocess a Dataset or the Stream guide to learn how to preprocess an IterableDataset.