Trọng tải

Your data can be stored in various places; they can be on your local machine's disk, in a
Kho lưu trữ Github và trong các cấu trúc dữ liệu trong bộ nhớ như từ điển Python và Pandas
DataFrames. Bất cứ nơi nào tập dữ liệu được lưu trữ,  Tập dữ liệu có thể giúp bạn tải tập dữ liệu đó.

Hướng dẫn này sẽ chỉ cho bạn cách tải tập dữ liệu từ:

- Trung tâm ôm mặt
- Tệp cục bộ
- Dữ liệu trong bộ nhớ
- Ngoại tuyến
- Một lát cắt cụ thể của phần tách

Để biết thêm chi tiết cụ thể về việc tải các phương thức tập dữ liệu khác, hãy xem âm thanh tải
hướng dẫn tập dữ liệu, hướng dẫn tải tập dữ liệu hình ảnh, hướng dẫn tải tập dữ liệu video hoặc tải văn bản
hướng dẫn tập dữ liệu.

Ôm mặt Hub

Bạn cũng có thể tải tập dữ liệu từ bất kỳ kho lưu trữ tập dữ liệu nào trên Hub! Bắt đầu bằng cách tạo một
dataset repository and upload your data files. Now you can use the load_dataset() function to
tải tập dữ liệu.

Ví dụ: hãy thử tải các tệp từ kho lưu trữ demo này bằng cách cung cấp kho lưu trữ
không gian tên và tên tập dữ liệu. Kho lưu trữ dữ liệu này chứa các tệp CSV và mã bên dưới
tải tập dữ liệu từ các tệp CSV:

```
>>> from datasets import load_dataset
>>> dataset = load_dataset("lhoestq/demo1")
```

Một số bộ dữ liệu có thể có nhiều phiên bản dựa trên thẻ Git, nhánh hoặc cam kết.
Sử dụng tham số sửa đổi để chỉ định phiên bản tập dữ liệu bạn muốn tải:

```
>>> dataset = load_dataset(
..."lhoestq/custom_squad",
...revision="main"# tag name, or branch name, or commit hash
... )
```

[!TIP]
Tham khảo hướng dẫn Tải tập dữ liệu lên Hub để biết thêm chi tiết về cách tạo
kho dữ liệu trên Hub và cách tải lên các tệp dữ liệu của bạn.

Theo mặc định, một tập dữ liệu sẽ tải tất cả dữ liệu vào phần tách tàu hoặc kiểm tra các đề cập hoặc phần tá
names in the data files names (e.g. "train", "test" and "validation"). Use the  data_files
tham số để ánh xạ các tệp dữ liệu thành các phần tách như train , validation và test :

```
>>> data_files = {"train": "train.csv", "test": "test.csv"}
>>> dataset = load_dataset("namespace/your_dataset_name", data_files=data_files)
```

[!WARNING]
If you don't specify which data files to use, load_dataset() will return all the data files. This
có thể mất nhiều thời gian nếu bạn tải một tập dữ liệu lớn như C4, dung lượng khoảng 13TB
dữ liệu.

Bạn cũng có thể tải một tập hợp con cụ thể của các tệp có tham số data_files hoặc data_dir.
Các tham số này có thể chấp nhận một đường dẫn tương đối phân giải thành đường dẫn cơ sở tương ứng vớ
nơi tập dữ liệu được tải từ đó.

```
>>> from datasets import load_dataset

# tải các tập tin phù hợp với mẫu grep
>>> c4_subset = load_dataset("allenai/c4", data_files="en/c4-train.0000*-of-01024.json.gz")

# tải tập dữ liệu từ thư mục en trên Hub
>>> c4_subset = load_dataset("allenai/c4", data_dir="en")
```

Tham số phân tách cũng có thể ánh xạ tệp dữ liệu tới một phần phân tách cụ thể:

```
>>> data_files = {"validation": "en/c4-validation.*.json.gz"}
>>> c4_validation = load_dataset("allenai/c4", data_files=data_files, split="validation")
```

Các tập tin cục bộ và từ xa

Bộ dữ liệu có thể được tải từ các tệp cục bộ được lưu trữ trên máy tính của bạn và từ các tệp từ xa. các datasets are most likely stored as a `csv`, `json`, `txt` or `parquet` file. The load_dataset() có thể tải từng loại tệp này.

CSV

Datasets can read a dataset made up of one or several CSV files (in this case, pass your CSV files as a list):

```
>>> from datasets import load_dataset
>>> dataset = load_dataset("csv", data_files="my_file.csv")
```

[!TIP]
Để biết thêm chi tiết, hãy xem cách tải tập dữ liệu dạng bảng từ hướng dẫn tệp CSV.

JSON

JSON files are loaded directly with load_dataset() as shown below:

```
>>> from datasets import load_dataset
>>> dataset = load_dataset("json", data_files="my_file.json")
```

Tệp JSON có định dạng đa dạng nhưng chúng tôi cho rằng định dạng hiệu quả nhất là có nhiều định dạng JSON objects; each line represents an individual row of data. For example:

```
{"a": 1, "b": 2.0, "c": "foo", "d": false}
{"a": 4, "b": -5.5, "c": null, "d": true}
```

Một định dạng JSON khác mà bạn có thể gặp là trường lồng nhau, trong trường hợp đó bạn cần chỉ định đối số trường như sau:

```
{"version": "0.1.0",
  "data": [{"a": 1, "b": 2.0, "c": "foo", "d": false},
                {"a": 4, "b": -5.5, "c": null, "d": true}]
}
```

```
>>> from datasets import load_dataset
>>> dataset = load_dataset("json", data_files="my_file.json", field="data")
```

Để tải các tệp JSON từ xa qua HTTP, thay vào đó hãy chuyển các URL:

```
>>> base_url = "https://rajpurkar.github.io/SQuAD-explorer/dataset/"
>>> dataset = load_dataset("json", data_files={"train": base_url + "train-v1.1.json", "validation"
```

Mặc dù đây là những định dạng JSON phổ biến nhất nhưng bạn sẽ thấy các tập dữ liệu khác được định dạng khác nhau. Bộ dữ liệu nhận ra các định dạng khác này và sẽ dự phòng tương ứng trên
Phương pháp tải JSON của Python để xử lý chúng.

sàn gỗ

Tệp sàn được lưu trữ ở định dạng cột, không giống như các tệp dựa trên hàng như CSV. Bộ dữ liệu lớn có thể được lưu trữ trong tệp Parquet vì nó trả về truy vấn của bạn hiệu quả hơn và nhanh hơn.

Để tải tệp Parquet:

```
>>> from datasets import load_dataset
>>> dataset = load_dataset("parquet", data_files={'train': 'train.parquet', 'test': 'test.parquet'
```

Để tải các tệp Parquet từ xa qua HTTP, thay vào đó hãy chuyển các URL:

```
>>> base_url = "https://huggingface.co/datasets/wikimedia/wikipedia/resolve/main/20231101.ab/"
>>> data_files = {"train": base_url + "train-00000-of-00001.parquet"}
>>> wiki = load_dataset("parquet", data_files=data_files, split="train")
```

Mũi tên

Các tệp mũi tên được lưu trữ ở định dạng cột trong bộ nhớ, không giống như các định dạng dựa trên hàng n

và các định dạng không nén như Parquet.

Để tải tệp Mũi tên:

```
>>> from datasets import load_dataset
>>> dataset = load_dataset("arrow", data_files={'train': 'train.arrow', 'test': 'test.arrow'})
```

Để tải các tệp Mũi tên từ xa qua HTTP, thay vào đó hãy chuyển các URL:

```
>>> base_url = "https://huggingface.co/datasets/croissantllm/croissant_dataset/resolve/main/englis
>>> data_files = {"train": base_url + "train/data-00000-of-00080.arrow"}
>>> wiki = load_dataset("arrow", data_files=data_files, split="train")
```

Mũi tên là định dạng tệp được sử dụng bởi  Bộ dữ liệu nâng cao, do đó bạn có thể tải cục bộ
Arrow file using Dataset.from_file() directly:

```
>>> from datasets import Dataset
>>> dataset = Dataset.from_file("data.arrow")
```

Unlike load_dataset(), Dataset.from_file() memory maps the Arrow file without preparing the
tập dữ liệu trong bộ đệm, giúp bạn tiết kiệm dung lượng ổ đĩa.
Thư mục cache để lưu kết quả xử lý trung gian sẽ là thư mục file Arrow trong
trường hợp đó.

For now only the Arrow streaming format is supported. The Arrow IPC file format (also known
as Feather V2) is not supported.

tập tin HDF5

Các tệp HDF5 thường được sử dụng để lưu trữ lượng lớn dữ liệu số trong khoa học
tính toán và học máy. Tải tệp HDF5 bằng  Bộ dữ liệu cũng tương tự như tải
Tệp CSV:

```
>>> from datasets import load_dataset
>>> dataset = load_dataset("hdf5", data_files="data.h5")
```

Lưu ý rằng trình tải HDF5 giả định rằng tệp có cấu trúc "dạng bảng", tức là tất cả các tập dữ liệu trong the file have (the same number of) rows on their first dimension.

SQL

Read database contents with from_sql() by specifying the URI to connect to your database.
Bạn có thể đọc cả tên bảng và truy vấn:

```
>>> from datasets import Dataset
# tải toàn bộ bảng
>>> dataset = Dataset.from_sql("data_table_name", con="sqlite:///sqlite_file.db")
# tải từ truy vấn
>>> dataset = Dataset.from_sql("SELECT text FROM table WHERE length(text) > 100 LIMIT 10", con="sq
```

[!TIP]
Để biết thêm chi tiết, hãy xem cách tải tập dữ liệu dạng bảng từ hướng dẫn cơ sở dữ liệu SQL.

Bộ dữ liệu Web

Định dạng WebDataset dựa trên kho lưu trữ TAR và phù hợp với các tập dữ liệu hình ảnh lớn.
Because of their size, WebDatasets are generally loaded in streaming mode (using streaming=True ).

Bạn có thể tải WebDataset như thế này:

```
>>> from datasets import load_dataset
>>>
>>> path = "path/to/train/*.tar"
>>> dataset = load_dataset("webdataset", data_files={"train": path}, split="train", streaming=True
```

Để tải WebDatasets từ xa qua HTTP, thay vào đó hãy chuyển các URL:

```
>>> from datasets import load_dataset
>>>
>>> base_url = "https://huggingface.co/datasets/lhoestq/small-publaynet-wds/resolve/main/publaynet
>>> urls = [base_url.format(i=i) for i in range(4)]
>>> dataset = load_dataset("webdataset", data_files={"train": urls}, split="train", streaming=True
```

Đa xử lý

When a dataset is made of several files (that we call "shards"), it is possible to significantly
tăng tốc bước tải xuống và chuẩn bị dữ liệu.

Bạn có thể chọn số lượng quy trình bạn muốn sử dụng để chuẩn bị tập dữ liệu song song bằng cách sử dụng
num_proc .
Trong trường hợp này, mỗi quy trình được cung cấp một tập hợp con các phân đoạn để chuẩn bị:

từ tập dữ liệu nhập Load_dataset

```
imagenet = load_dataset("timm/imagenet-1k-wds", num_proc=8)
ml_librispeech_spanish = load_dataset("facebook/multilingual_librispeech", "spanish", num_proc=8)
```

Dữ liệu trong bộ nhớ

Bộ dữ liệu cũng sẽ cho phép bạn tạo Bộ dữ liệu trực tiếp từ các cấu trúc dữ liệu trong bộ nhớ như
Từ điển Python và khung dữ liệu Pandas.

Từ điển Python

Load Python dictionaries with from_dict():

```
>>> from datasets import Dataset
>>> my_dict = {"a": [1, 2, 3]}
>>> dataset = Dataset.from_dict(my_dict)
```

Danh sách từ điển Python

Load a list of Python dictionaries with  from_list() :

```
>>> from datasets import Dataset
>>> my_list = [{"a": 1}, {"a": 2}, {"a": 3}]
>>> dataset = Dataset.from_list(my_list)
```

Trình tạo Python

Create a dataset from a Python generator with from_generator():

```
>>> from datasets import Dataset
>>> def my_gen():
...for i in range(1, 4):
...yield {"a": i}
None
>>> dataset = Dataset.from_generator(my_gen)
```

Cách tiếp cận này hỗ trợ tải dữ liệu lớn hơn bộ nhớ khả dụng.

Bạn cũng có thể xác định tập dữ liệu được phân chia bằng cách chuyển danh sách tới gen_kwargs :

```
>>> def gen(shards):
...cho phân đoạn trong phân đoạn:
...with open(shard) as f:
...cho dòng trong f:
...yield {"line": line}
None
>>> shards = [f"data{i}.txt" for i in range(32)]
>>> ds = IterableDataset.from_generator(gen, gen_kwargs={"shards": shards})
>>> ds = ds.shuffle(seed=42, buffer_size=10_000)# shuffles the shards order + uses a shuffle buf
>>> from torch.utils.data import DataLoader
>>> dataloader = DataLoader(ds.with_format("torch"), num_workers=4)# give each worker a subset o
```

Khung dữ liệu gấu trúc

Load Pandas DataFrames with from_pandas():

```
>>> from datasets import Dataset
>>> import pandas as pd
>>> df = pd.DataFrame({"a": [1, 2, 3]})
>>> dataset = Dataset.from_pandas(df)
```

[!TIP]
Để biết thêm chi tiết, hãy xem cách tải tập dữ liệu dạng bảng từ Pandas DataFrames

hướng dẫn.

Ngoại tuyến

Ngay cả khi bạn không có kết nối Internet, bạn vẫn có thể tải tập dữ liệu. Miễn là
trước đây bạn đã tải xuống tập dữ liệu từ kho lưu trữ Hub, tập dữ liệu đó sẽ được lưu vào bộ nhớ đệm. Điều
bạn có thể tải lại tập dữ liệu từ bộ đệm và sử dụng ngoại tuyến.

Nếu bạn biết mình sẽ không có quyền truy cập Internet, bạn có thể chạy  Bộ dữ liệu ở chế độ ngoại tuyến ho
tiết kiệm thời gian vì thay vì phải chờ tải xuống Trình tạo tập dữ liệu hết thời gian,
Bộ dữ liệu sẽ nhìn trực tiếp vào bộ đệm. Đặt biến môi trường HF_HUB_OFFLINE thành 1 thành
bật chế độ ngoại tuyến hoàn toàn.

Chia lát

Bạn cũng có thể chọn chỉ tải các lát cắt cụ thể của phần tách. Có hai lựa chọn để cắt một
tách: sử dụng chuỗi hoặc API ReadInstruction. Chuỗi nhỏ gọn hơn và dễ đọc hơn
các trường hợp đơn giản, trong khi ReadInstruction dễ sử dụng hơn với các tham số cắt có thể thay đổi.

Ghép nối một đoàn tàu và phân chia thử nghiệm theo:

```
>>> train_test_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="train+
===STRINGAPI-READINSTRUCTION-SPLIT===
>>> ri = datasets.ReadInstruction("train") + datasets.ReadInstruction("test")
>>> train_test_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split=ri)
```

Chọn các hàng cụ thể của đoàn tàu:

```
>>> train_10_20_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="train
===STRINGAPI-READINSTRUCTION-SPLIT===
>>> train_10_20_ds = datasets.load_dataset("bookcorpu", split=datasets.ReadInstruction("train", fr
```

Hoặc chọn tỷ lệ phần trăm của phần chia với:

```
>>> train_10pct_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="train
===STRINGAPI-READINSTRUCTION-SPLIT===
>>> train_10_20_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split=datase
```

Chọn sự kết hợp của tỷ lệ phần trăm từ mỗi phần chia:

```
>>> train_10_80pct_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="tr
===STRINGAPI-READINSTRUCTION-SPLIT===
>>> ri = (datasets.ReadInstruction("train", to=10, unit="%") + datasets.ReadInstruction("train", f
>>> train_10_80pct_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split=ri)
```

Cuối cùng, bạn thậm chí có thể tạo các phần tách được xác thực chéo. Ví dụ dưới đây tạo ra đường chéo 10 l
sự phân chia được xác thực. Mỗi tập dữ liệu xác thực là một đoạn 10% và tập dữ liệu huấn luyện tạo nên
90% phần bổ sung còn lại:

```
>>> val_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split=[f"train[{k}%:
>>> train_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split=[f"train[:{k
===STRINGAPI-READINSTRUCTION-SPLIT===
>>> val_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", [datasets.ReadInstru
>>> train_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", [(datasets.ReadIns
```

Phần trăm cắt và làm tròn

Hành vi mặc định là làm tròn các ranh giới đến số nguyên gần nhất cho các tập dữ liệu trong đó
ranh giới lát cắt được yêu cầu không chia đều cho 100. Như được hiển thị bên dưới, một số lát cắt có thể
chứa nhiều ví dụ hơn những ví dụ khác. Ví dụ: nếu phần chia đoàn tàu sau đây bao gồm 999
hồ sơ thì:

```
# 19 records, from 500 (included) to 519 (excluded).
>>> train_50_52_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="train
# 20 records, from 519 (included) to 539 (excluded).
>>> train_52_54_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="train
```

Nếu bạn muốn chia nhỏ có kích thước bằng nhau, thay vào đó hãy sử dụng cách làm tròn pct1_dropremaind
ranh giới phần trăm được chỉ định là bội số của 1%.

```
# 18 records, from 450 (included) to 468 (excluded).
>>> train_50_52pct1_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split=da
# 18 records, from 468 (included) to 486 (excluded).
>>> train_52_54pct1_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split=da
# Hoặc tương đương:
>>> train_50_52pct1_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="t
>>> train_52_54pct1_ds = datasets.load_dataset("ajibawa-2023/General-Stories-Collection", split="t
```

[!WARNING]
Việc làm tròn pct1_dropremainder có thể cắt bớt các mẫu cuối cùng trong tập dữ liệu nếu số
ví dụ trong tập dữ liệu của bạn không chia đều cho 100.

Khắc phục sự cố

Đôi khi, bạn có thể nhận được kết quả không mong muốn khi tải tập dữ liệu. Hai trong số nhiều nhất
các vấn đề phổ biến bạn có thể gặp phải là tải xuống tập dữ liệu theo cách thủ công và chỉ định
các tính năng của một tập dữ liệu.

Chỉ định tính năng

Khi bạn tạo tập dữ liệu từ các tệp cục bộ, các Tính năng sẽ được Apache tự động suy ra
Mũi tên. Tuy nhiên, các tính năng của tập dữ liệu có thể không phải lúc nào cũng phù hợp với mong đợi của
có thể muốn tự xác định các tính năng. Ví dụ sau đây cho thấy cách bạn có thể thêm
nhãn tùy chỉnh với tính năng ClassLabel.

Bắt đầu bằng cách xác định nhãn của riêng bạn với lớp Tính năng:

```
>>> class_names = ["sadness", "joy", "love", "anger", "fear", "surprise"]
>>> emotion_features = Features({'text': Value('string'), 'label': ClassLabel(names=class_names)})
```

Next, specify the  features parameter in load_dataset() with the features you just created:

```
>>> dataset = load_dataset('csv', data_files=file_dict, delimiter=';', column_names=['text', 'labe
```

Bây giờ, khi bạn xem các tính năng của tập dữ liệu, bạn có thể thấy nó sử dụng các nhãn tùy chỉnh mà bạn
định nghĩa:

```
>>> dataset['train'].features
{'text': Value('string'),
'label': ClassLabel(names=['sadness', 'joy', 'love', 'anger', 'fear', 'surprise'])}
```