Các lớp chính

DatasetInfodatasets.DatasetInfo

tập dữ liệu lớp.DatasetInfodatasets.DatasetInfohttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/info.py#L92[{"name": "description", "val": ": str = "}, {"name": "citation", "val": ": str = "}, {"name": "homepage", "val": ": str = "}, {"name": "license", "val": ": str = "}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "post_processed", "val": ": typing.Optional[datasets.info.PostProcessedInfo] = None"}, {"name": "supervised_keys", "val": ": typing.Optional[datasets.info.SupervisedKeysData] = None"}, {"name": "builder_name", "val": ": typing.Optional[str] = None"}, {"name": "dataset_name", "val": ": typing.Optional[str] = None"}, {"name": "config_name", "val": ": typing.Optional[str] = None"}, {"name": "version", "val": ": typing.Union[str, datasets.utils.version.Version, NoneType] = None"}, {"name": "splits", "val": ": typing.Optional[dict] = None"}, {"name": "download_checksums", "val": ": typing.Optional[dict] = None"}, {"name": "download_size", "val": ": typing.Optional[int] = None"}, {"name": "post_processing_size", "val": ": typing.Optional[int] = None"}, {"name": "dataset_size", "val": ": typing.Optional[int] = None"}, {"name": "size_in_bytes", "val": ": typing.Optional[int] = None"}]- description (str) --
Một mô tả của tập dữ liệu.

- citation (str) --
  Một trích dẫn BibTeX của tập dữ liệu.
- homepage (str) --
  URL tới trang chủ chính thức của tập dữ liệu.
- license (str) --
  Giấy phép của tập dữ liệu. Nó có thể là tên của giấy phép hoặc một đoạn văn có chứa điều khoản của giấy phép.
- features (Features, optional) --
  Các tính năng được sử dụng để chỉ định các loại cột của tập dữ liệu.
- post_processed (PostProcessedInfo , optional) --
  Thông tin liên quan đến các tài nguyên có thể xử lý hậu kỳ của tập dữ liệu. Vì ví dụ, nó có thể chứa thông tin của một chỉ mục.
- supervised_keys (SupervisedKeysData , optional) --
  Chỉ định tính năng đầu vào và nhãn cho việc học có giám sát nếu áp dụng cho

dataset (legacy from TFDS).

- builder_name (str , optional) --
  Tên của lớp con GeneratorBasedBuilder được sử dụng để tạo tập dữ liệu. Nó cũng là phiên bản snake_case của tên lớp trình tạo tập dữ liệu.
- config_name (str , optional) --
  Tên của cấu hình bắt nguồn từ BuilderConfig.
- version (str or Version, optional) --
  Phiên bản của tập dữ liệu.
- splits (dict , optional) --
  Ánh xạ giữa tên phân chia và siêu dữ liệu.
- download_checksums (dict , optional) --
  Ánh xạ giữa URL để tải xuống tổng kiểm tra của tập dữ liệu và tương ứng siêu dữ liệu.
- download_size (int , optional) --
  Kích thước của tệp cần tải xuống để tạo tập dữ liệu, tính bằng byte.
- post_processing_size (int , optional) --
  Kích thước của tập dữ liệu tính bằng byte sau khi xử lý hậu kỳ, nếu có.
- dataset_size (int , optional) --
  Kích thước kết hợp tính bằng byte của bảng Mũi tên cho tất cả các phần tách.
- size_in_bytes (int , optional) --
  The combined size in bytes of all files associated with the dataset (downloaded files + Arrow files).
- **config_kwargs (additional keyword arguments) --
  Đối số từ khóa sẽ được chuyển đến BuilderConfig và được sử dụng trong DatasetBuilder.0
  Thông tin về một tập dữ liệu.

DatasetInfo ghi lại các bộ dữ liệu, bao gồm tên, phiên bản và tính năng của nó.
Xem các đối số và thuộc tính của hàm tạo để biết danh sách đầy đủ.

Không phải tất cả các lĩnh vực đều được biết đến trong quá trình xây dựng và có thể được cập nhật sau.

from_directorydatasets.DatasetInfo.from_directoryhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/info.py#L247[{"name": "dataset_info_dir", "val": ": str"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}]- dataset_info_dir (str) --
Thư mục chứa tệp siêu dữ liệu. Cái này
phải là thư mục gốc của một phiên bản tập dữ liệu cụ thể.

- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0
  Tạo DatasetInfo từ tệp JSON trong tập dữ liệu_info_dir.

This function updates all the dynamically generated fields (num_examples,
hash, time of creation,...) of the DatasetInfo.

Điều này sẽ ghi đè lên tất cả siêu dữ liệu trước đó.

Ví dụ:

```
>>> from datasets import DatasetInfo
>>> ds_info = DatasetInfo.from_directory("/path/to/directory/")
```

write_to_directorydatasets.DatasetInfo.write_to_directoryhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/info.py#L186[{"name": "dataset_info_dir", "val": ""}, {"name": "pretty_print", "val": " = False"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}]- dataset_info_dir (str) --
Thư mục đích.

- pretty_print (bool , defaults to False) --
  Nếu True, JSON sẽ được in đẹp với mức thụt lề là 4.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0
  Write DatasetInfo and license (if present) as JSON files to dataset_info_dir .

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.info.write_to_directory("/path/to/directory/")
```

Tập dữ liệudatasets.Dataset

Bộ dữ liệu của lớp cơ sở triển khai Bộ dữ liệu được hỗ trợ bởi bảng Mũi tên Apache.

bộ dữ liệu lớp.Datasetdatasets.Datasethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L695[{"name": "arrow_table", "val": ": Table"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "indices_table", "val": ": typing.Optional[datasets.table.Table] = None"}, {"name": "fingerprint", "val": ": typing.Optional[str] = None"}]
Bộ dữ liệu được hỗ trợ bởi bảng Mũi tên.

add_columndatasets.Dataset.add_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L6059[{"name": "name", "val": ": str"}, {"name": "column", "val": ": typing.Union[list, numpy.ndarray]"}, {"name": "new_fingerprint", "val": ": str"}, {"name": "feature", "val": ": typing.Union[dict, list, tuple, datasets.features.features.Value, bộ dữ liệu.features.features.ClassLabel, bộ dữ liệu.features.translation.Translation, bộ dữ liệu.features.translation.TranslationVariableLanguages, bộ dữ liệu.features.features.LargeList, bộ dữ liệu.features.features.List, bộ dữ liệu.features.features.Array2D, bộ dữ liệu.features.features.Array3D, bộ dữ liệu.features.features.Array4D, bộ dữ liệu.features.features.Array5D, bộ dữ liệu.features.audio.Audio, bộ dữ liệu.features.image.Image, bộ dữ liệu.features.video.Video, datasets.features.pdf.Pdf, NoneType] = None"}]- name (str) --
Tên cột.

- column (list or np.array) --
  Dữ liệu cột sẽ được thêm vào.
- feature (FeatureType or None , defaults to None) --
  Kiểu dữ liệu cột.0Dataset
  Thêm cột vào Tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> more_text = ds["text"]
>>> ds = ds.add_column(name="text_2", column=more_text)
>>> ds
Dataset({
    features: ['text', 'label', 'text_2'],
    num_rows: 1066
})
```

add_itemdatasets.Dataset.add_itemhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L6317[{"name": "item", "val": ": dict"}, {"name": "new_fingerprint", "val": ": str"}]- item (dict) --
Dữ liệu mục cần được thêm vào.0Dataset
Thêm mục vào Tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> new_review = {'label': 0, 'text': 'this movie is the absolute worst thing I have ever seen'}
>>> ds = ds.add_item(new_review)
>>> ds[-1]
{'label': 0, 'text': 'this movie is the absolute worst thing I have ever seen'}
```

from_filedatasets.Dataset.from_filehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L778[{"name": "filename", "val": ": str"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "indices_filename", "val": ": typing.Optional[str] = None"}, {"name": "in_memory", "val": ": bool = False"}]- filename (str) --
Tên tệp của tập dữ liệu.

- info (DatasetInfo , optional) --
  Thông tin về tập dữ liệu, như mô tả, trích dẫn, v.v.
- split (NamedSplit , optional) --
  Tên của tập dữ liệu được chia.
- indices_filename (str , optional) --

Tên tập tin của các chỉ mục.

- in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.0Dataset
  Khởi tạo Tập dữ liệu được hỗ trợ bởi bảng Mũi tên ở tên tệp.

from_bufferdatasets.Dataset.from_bufferhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L818[{"name": "buffer", "val": ": Buffer"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "indices_buffer", "val": ": typing.Optional[pyarrow.lib.Buffer] = None"}]- buffer (pyarrow.Buffer) --
Bộ đệm mũi tên.

- info (DatasetInfo , optional) --
  Thông tin về tập dữ liệu, như mô tả, trích dẫn, v.v.
- split (NamedSplit , optional) --
  Tên của tập dữ liệu được chia.
- indices_buffer (pyarrow.Buffer , optional) --
  Chỉ số Mũi tên buffer.0Dataset
  Khởi tạo Tập dữ liệu được hỗ trợ bởi bộ đệm Mũi tên.

from_pandasdatasets.Dataset.from_pandashttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L850[{"name": "df", "val": ": DataFrame"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "preserve_index", "val": ": typing.Optional[bool] = None"}]- df (pandas.DataFrame) --
Khung dữ liệu chứa tập dữ liệu.

- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- info (DatasetInfo , optional) --
  Thông tin về tập dữ liệu, như mô tả, trích dẫn, v.v.
- split (NamedSplit , optional) --
  Tên của tập dữ liệu được chia.
- preserve_index (bool , optional) --
  Có lưu trữ chỉ mục dưới dạng cột bổ sung trong Tập dữ liệu kết quả hay không.

Giá trị mặc định của Không có sẽ lưu trữ chỉ mục dưới dạng một cột, ngoại trừ RangeIndex là chỉ được lưu trữ dưới dạng siêu dữ liệu.
Use preserve_index=True to force it to be stored as a column.0Dataset

Chuyển đổi pandas.DataFrame thành pyarrow.Table để tạo Tập dữ liệu.

Các kiểu cột trong Bảng Mũi tên thu được được suy ra từ các kiểu dtype của
pandas.Series trong
DataFrame. Trong trường hợp Sê-ri không phải đối tượng, dtype NumPy được dịch sang Mũi tên của nó tương đương. trong
trường hợp object , chúng ta cần đoán kiểu dữ liệu bằng cách xem các đối tượng Python trong Series này.

Xin lưu ý rằng Chuỗi đối tượng dtype không mang đủ thông tin để luôn dẫn đến
mũi tên ý nghĩa
kiểu. Trong trường hợp chúng tôi không thể suy ra một loại, ví dụ: vì DataFrame có độ dài 0 hoặc Chỉ loạt phim
chứa các đối tượng None/nan, loại được đặt thành null . Hành vi này có thể tránh được bằng cách xây dựng rõ ràng
các tính năng và chuyển nó đến chức năng này.

Important: a dataset created with from_pandas() lives in memory
và do đó không có thư mục bộ đệm liên quan.
Điều này có thể thay đổi trong tương lai, nhưng trong lúc đó nếu bạn
muốn giảm mức sử dụng bộ nhớ, bạn nên ghi lại vào đĩa
và tải lại bằng cách sử dụng ví dụ: save_to_disk/load_from_disk.

Ví dụ:

```
>>> ds = Dataset.from_pandas(df)
```

from_dictdatasets.Dataset.from_dicthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L964[{"name": "mapping", "val": ": dict"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}]- mapping (Mapping) --
Ánh xạ các chuỗi vào danh sách Mảng hoặc Python.

- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- info (DatasetInfo , optional) --
  Thông tin về tập dữ liệu, như mô tả, trích dẫn, v.v.
- split (NamedSplit , optional) --
  Tên của tập dữ liệu chia.0Dataset

Chuyển đổi dict thành pyarrow.Table để tạo Bộ dữ liệu.

Important: a dataset created with from_dict() lives in memory
và do đó không có thư mục bộ đệm liên quan.
Điều này có thể thay đổi trong tương lai, nhưng trong lúc đó nếu bạn
muốn giảm mức sử dụng bộ nhớ, bạn nên ghi lại vào đĩa
và tải lại bằng cách sử dụng ví dụ: save_to_disk/load_from_disk.

from_generatordatasets.Dataset.from_generatorhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1114[{"name": "generator", "val": ": typing.Callable"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "gen_kwargs", "val": ": typing.Optional[dict] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "split", "val": ": NamedSplit = NamedSplit('train')"}, {"name": "**kwargs", "val": ""}]- generator (-- Callable):
Một hàm tạo tạo ra các ví dụ.

- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str , optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- gen_kwargs(dict , optional) --
  Đối số từ khóa được chuyển đến trình tạo có thể gọi được.
  Bạn có thể xác định tập dữ liệu được phân đoạn bằng cách chuyển danh sách các phân đoạn trong gen_
  num_proc lớn hơn 1.
- num_proc (int , optional, defaults to None) --
  Số lượng quy trình khi tải xuống và tạo tập dữ liệu cục bộ.

Điều này hữu ích nếu tập dữ liệu được tạo thành từ nhiều tệp. Đa xử lý bị vô hiệu hóa bởi mặc định.

Nếu num_proc lớn hơn 1 thì tất cả giá trị danh sách trong gen_kwargs phải giống nhau chiều dài. Các giá trị này sẽ được phân chia giữa các cuộc gọi đến trình tạo. Số lượng mảnh vỡ sẽ là mức tối thiểu trong danh sách ngắn nhất trong gen_kwargs và num_proc .

- split (NamedSplit, defaults to Split.TRAIN) --

  Tên phân chia được gán cho tập dữ liệu.

- **kwargs (additional keyword arguments) --

  Đối số từ khóa được chuyển tới: GeneratorConfig .0Dataset

  Tạo Bộ dữ liệu từ một trình tạo.

Ví dụ:

```
>>> def gen():
...yield {"text": "Good", "label": 0}
...yield {"text": "Bad", "label": 1}
None
>>> ds = Dataset.from_generator(gen)
```

```
>>> 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 = Dataset.from_generator(gen, gen_kwargs={"shards": shards})
```

datadatasets.Dataset.datahttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1808[]

Bảng Mũi tên Apache hỗ trợ tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.data
Bảng ánh xạ bộ nhớ
văn bản: chuỗi
nhãn: int64
None
text: [["compassionately explores the seemingly irreconcilable situation between conservative chri
label: [[1,1,1,1,1,1,1,1,1,1,...,0,0,0,0,0,0,0,0,0,0]]
```

cache_filesdatasets.Dataset.cache_fileshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1828[]
Các tệp bộ đệm chứa bảng Mũi tên Apache sao lưu tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.cache_files
[{'filename': '/root/.cache/huggingface/datasets/rotten_tomatoes_movie_review/default/1.0.0/40d411
```

num_columnsdatasets.Dataset.num_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1846[]
Số cột trong tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.num_columns
2
```

num_rowsdatasets.Dataset.num_rowshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1861[]
Number of rows in the dataset (same as Dataset.len()).

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.num_rows
1066
```

cột_namesdatasets.Dataset.column_nameshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1878[]
Tên của các cột trong tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.column_names
['text', 'label']
```

Shapedatasets.Dataset.shapehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1893[]
Shape of the dataset (number of columns, number of rows).

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.shape
(1066, 2)
```

Uniquedatasets.Dataset.uniquehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1910[{"name": "column", "val": ": str"}]- column (str) --
Column name (list all the column names with column_names).0 list List of unique elements in cột đã cho.
Trả về danh sách các phần tử duy nhất trong một cột.

Điều này được thực hiện ở phần phụ trợ cấp thấp và như vậy rất nhanh.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.unique('label')
[1, 0]
```

Flattendatasets.Dataset.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2016[{"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}, {"name": "max_depth", "val": " = 16"}]- new_fingerprint (str , optional) --
Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số.0DatasetMột bản sao của tập dữ liệu với các cột được làm phẳng.
Làm phẳng bàn.
Mỗi cột có kiểu cấu trúc được làm phẳng thành một cột trên mỗi trường cấu trúc.
Các cột khác không thay đổi.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("rajpurkar/squad", split="train")
>>> ds.features
{'id': Value('string'),
 'title': Value('string'),
 'context': Value('string'),
 'question': Value('string'),
 'answers': {'text': List(Value('string')),
 'answer_start': List(Value('int32'))}}
>>> ds = ds.flatten()
>>> ds
Dataset({
    features: ['id', 'title', 'context', 'question', 'answers.text', 'answers.answer_start'],
    num_rows: 87599
})
```

castdatasets.Dataset.casthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2063[{"name": "features", "val": ": Features"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "cache_file_name",

"val": ": typing.Optional[str] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}]- features (Features) --
Các tính năng mới để truyền tập dữ liệu tới.
Tên của các trường trong đối tượng phải khớp với tên cột hiện tại.
Loại dữ liệu cũng phải có thể chuyển đổi từ loại này sang loại khác.
For non-trivial conversion, e.g. str <-> ClassLabel you should use map() to update the
Tập dữ liệu.

- batch_size (int , defaults to 1000) --
  Số lượng mẫu mỗi lô được cung cấp để truyền.
  If batch_size <= 0 or batch_size == None then provide the full dataset as a single batch
  để đúc.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- load_from_cache_file (bool , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ tính toán hiện tại từ hàm
  có thể được xác định, hãy sử dụng nó thay vì tính toán lại.
- cache_file_name (str , optional, defaults to None) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  kết quả tính toán thay vì tên tệp bộ đệm được tạo tự động.
- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
  tốc độ xử lý.
  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn
  temporary memory while running map().
- num_proc (int , optional, defaults to None) --
  Số lượng tiến trình cho đa xử lý. Theo mặc định thì không
  sử dụng multiprocessing.0DatasetMột bản sao của tập dữ liệu với các tính năng được truyền.

Truyền tập dữ liệu sang một bộ tính năng mới.

Ví dụ:

```
>>> from datasets import load_dataset, ClassLabel, Value
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
>>> new_features = ds.features.copy()
>>> new_features['label'] = ClassLabel(names=['bad', 'good'])
>>> new_features['text'] = Value('large_string')
>>> ds = ds.cast(new_features)
>>> ds.features
{'label': ClassLabel(names=['bad', 'good']),
 'text': Value('large_string')}
```

cast_columndatasets.Dataset.cast_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2147[{"name": "column", "val": ": str"}, {"name": "feature", "val": ": typing.Union[dict, list, tuple, datasets.features.features.Value, bộ dữ liệu.features.features.ClassLabel, bộ dữ liệu.features.translation.Translation, bộ dữ liệu.features.translation.TranslationVariableLanguages, bộ dữ liệu.features.features.LargeList, bộ dữ liệu.features.features.List, bộ dữ liệu.features.features.Array2D, bộ dữ liệu.features.features.Array3D, bộ dữ liệu.features.features.Array4D, bộ dữ liệu.features.features.Array5D, bộ dữ liệu.features.audio.Audio, bộ dữ liệu.features.image.Image, bộ dữ liệu.features.video.Video, datasets.features.pdf.Pdf]"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]-

column (str) --

Tên cột.

- feature (FeatureType) --

  Tính năng mục tiêu.
- new_fingerprint (str , optional) --

  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.

  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số.0Dataset

  Cột truyền tới tính năng để giải mã.

Ví dụ:

```
>>> from datasets import load_dataset, ClassLabel
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
>>> ds = ds.cast_column('label', ClassLabel(names=['bad', 'good']))
>>> ds.features
{'label': ClassLabel(names=['bad', 'good']),
 'text': Value('string')}
```

Remove_columnsdatasets.Dataset.remove_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2190[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]-column_names (Union[str, List[str]]) --
Name of the column(s) to remove.

- new_fingerprint (str , optional) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số.0DatasetMột bản sao của đối tượng tập dữ liệu không có cột cần xóa.

Remove one or several column(s) in the dataset and the features associated to them.

You can also remove a column using map() with remove_columns but the present method không sao chép dữ liệu của các cột còn lại và do đó nhanh hơn.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds = ds.remove_columns('label')
Dataset({
    features: ['text'],
    num_rows: 1066
})
>>> ds = ds.remove_columns(column_names=ds.column_names) # Removing all the columns returns an er
Dataset({
    features: [],
    num_rows: 0
})
```

đổi tên_columndatasets.Dataset.rename_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2245[{"name": "original_column_name", "val": ": str"}, {"name": "new_column_name", "val": ": str"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]- original_column_name (str) --
Tên cột cần đổi tên.

- new_column_name (str) --
  Tên mới cho cột.
- new_fingerprint (str , optional) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số.0DatasetMột bản sao của tập dữ liệu có cột được đổi tên.

Đổi tên một cột trong tập dữ liệu và di chuyển các tính năng được liên kết với cột ban đầu dưới cột mới
tên.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds = ds.rename_column('label', 'label_new')
Dataset({
    features: ['text', 'label_new'],
    num_rows: 1066
})
```

đổi tên_columnsdatasets.Dataset.rename_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2311[{"name": "column_mapping", "val": ": dict"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]- column_mapping (Dict[str, str]) --
Ánh xạ các cột để đổi tên thành tên mới

- new_fingerprint (str , optional) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số.0DatasetMột bản sao của tập dữ liệu với các cột được đổi tên

Đổi tên một số cột trong tập dữ liệu và di chuyển các tính năng được liên kết với bản gốc cột bên dưới
tên cột mới.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds = ds.rename_columns({'text': 'text_new', 'label': 'label_new'})
Dataset({
    features: ['text_new', 'label_new'],
    num_rows: 1066
})
```

select_columnsdatasets.Dataset.select_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2378[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]- column_names (Union[str, List[str]]) --

Name of the column(s) to keep.

- new_fingerprint (str , optional) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi. Nếu không có ,
  dấu vân tay mới được tính toán bằng cách sử dụng hàm băm của dấu vân tay trước đó
  dấu vân tay và các đối số biến đổi.0DatasetMột bản sao của đối tượng tập dữ liệu mà chỉ
  bao gồm
  các cột đã chọn.
  Select one or several column(s) in the dataset and the features
  liên quan đến họ.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds = ds.select_columns(['text'])
>>> ds
Dataset({
    features: ['text'],
    num_rows: 1066
})
```

class_encode_columndatasets.Dataset.class_encode_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1941[{"name": "column", "val": ": str"}, {"name": "include_nulls", "val": ": bool = False"}]- column (str) --
The name of the column to cast (list all the column names with column_names)

- include_nulls (bool , defaults to False) --
  Có bao gồm các giá trị null trong nhãn lớp hay không. Nếu True, giá trị null sẽ được mã hóa
  làm nhãn lớp "Không".
  0
  Truyền cột đã cho dưới dạng ClassLabel và cập nhật bảng.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("boolq", split="validation")
>>> ds.features
{'answer': Value('bool'),
 'passage': Value('string'),
 'question': Value('string')}
>>> ds = ds.class_encode_column('answer')
>>> ds.features
{'answer': ClassLabel(num_classes=2, names=['False', 'True']),
 'passage': Value('string'),
 'question': Value('string')}
```

lendatasets.Dataset.lenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2434[]
Số hàng trong tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.__len__
<bound method Dataset.__len__ of Dataset({
    features: ['text', 'label'],
    num_rows: 1066
})>
```

iterdatasets.Dataset.iterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2451[]
Lặp lại thông qua các ví dụ.

If a formatting is set with Dataset.set_format() rows will be returned with the
định dạng đã chọn.

iterdatasets.Dataset.iterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2480[{"name": "batch_size", "val": ": int"}, {"name": "drop_last_batch", "val": ": bool = False"}]- batch_size (int) -- size of each batch to yield.

- drop_last_batch (bool , default False) -- Whether a last batch smaller than the

batch_size nên là
đánh rơi0
Lặp lại qua các lô có kích thước batch_size.

If a formatting is set with [~datasets.Dataset.set_format] rows will be returned with the định dạng đã chọn.

formatted_asdatasets.Dataset.formatted_ashttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2524[{"name": "type", "val": ": typing.Optional[str] = None"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}, {"name": "**format_kwargs", "val": ""}]- type (str , optional) --
Loại đầu ra được chọn trong
[None, 'numpy', 'torch', 'tensorflow', 'jax', 'arrow', 'pandas', 'polars'] .
None means `getitem`` returns python objects (default).

- columns (List[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  None means __getitem__ returns all columns (default).
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
- **format_kwargs (additional keyword arguments) --
  Các đối số từ khóa được truyền cho hàm chuyển đổi như np.array , torch.tensor hoặc tenorflow.ragged.constant .0
  To be used in a with statement. Set __getitem__ return format (type and columns).

set_formatdatasets.Dataset.set_formathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2556[{"name": "type", "val": ": typing.Optional[str] = None"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}, {"name": "**format_kwargs", "val": ""}]- type (str , optional) --
Loại đầu ra được chọn trong
[None, 'numpy', 'torch', 'tensorflow', 'jax', 'arrow', 'pandas', 'polars'] .
None means __getitem__ returns python objects (default).

- columns (List[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  None means __getitem__ returns all columns (default).

- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
- **format_kwargs (additional keyword arguments) --
  Các đối số từ khóa được truyền cho hàm chuyển đổi như np.array , torch.tensor hoặc tensorflow.ragged.constant .0
  Set __getitem__ return format (type and columns). The data formatting is applied on-the-bay.
  The format type (for example "numpy") is used to format batches when using __getitem__ .
  It's also possible to use custom transforms for formatting using set_transform().

It is possible to call map() after calling set_format . Since map may add new columns, then the danh sách các cột được định dạng

được cập nhật. Trong trường hợp này, nếu bạn áp dụng bản đồ trên tập dữ liệu để thêm cột mới thì điều này cột sẽ được định dạng là:

```
new formatted columns = (all columns - previously unformatted columns)
```

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> ds = ds.map(lambda x: tokenizer(x['text'], truncation=True, padding=True), batched=True)
>>> ds.set_format(type='numpy', columns=['text', 'label'])
>>> ds.format
{'type': 'numpy',
'format_kwargs': {},
'columns': ['text', 'label'],
'output_all_columns': False}
```

set_transformdatasets.Dataset.set_transformhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2664[{"name": "transform", "val": ": typing.Optional[typing.Callable]"}, {"name": "columns", "val": ": typing.Optional[list] = None"},

{"name": "output_all_columns", "val": ": bool = False"}]- transform (Callable , optional) --
User-defined formatting transform, replaces the format defined by set_format().
A formatting function is a callable that takes a batch (as a dict) as input and returns a batch.
Hàm này được áp dụng ngay trước khi trả về các đối tượng trong __getitem__ .

- columns (List[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  Nếu được chỉ định thì lô đầu vào của phép biến đổi chỉ chứa các cột đó.
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
  Nếu được đặt thành True thì các cột chưa được định dạng khác sẽ được giữ nguyên với đầu ra của biến đổi.0
  Đặt định dạng trả về __getitem__ bằng cách sử dụng phép biến đổi này. Việc chuyển đổi được áp dụng n các đợt khi __getitem__ được gọi.
  As set_format(), this can be reset using reset_format().

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> tokenizer = AutoTokenizer.from_pretrained('bert-base-uncased')
>>> def encode(batch):
...return tokenizer(batch['text'], padding=True, truncation=True, return_tensors='pt')
>>> ds.set_transform(encode)
>>> ds[0]
{'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1]),
  'input_ids': tensor([101, 29353,2135, 15102,1996,9428, 20868,2890,8663,6895,
                       20470,2571,3663,2090,4603,3017,3008,1998,2037, 24211,
                       5637,1998, 11690,2336,1012,102]),
  'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                       0, 0])}
```

reset_formatdatasets.Dataset.reset_formathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2635[]
Đặt lại định dạng trả về __getitem__ cho đối tượng python và tất cả các cột.

Same as self.set_format()

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> ds = ds.map(lambda x: tokenizer(x['text'], truncation=True, padding=True), batched=True)
>>> ds.set_format(type='numpy', columns=['input_ids', 'token_type_ids', 'attention_mask', 'label']
>>> ds.format
{'columns': ['input_ids', 'token_type_ids', 'attention_mask', 'label'],
  'format_kwargs': {},
  'output_all_columns': Sai,
  'type': 'numpy'}
>>> ds.reset_format()
>>> ds.format
{'columns': ['text', 'label', 'input_ids', 'token_type_ids', 'attention_mask'],
  'format_kwargs': {},
  'output_all_columns': Sai,
  'type': None}
```

with_formatdatasets.Dataset.with_formathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2707[{"name": "type", "val": ": typing.Optional[str] = None"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}, {"name": "**format_kwargs", "val": ""}]- type (str , optional) -- Loại đầu ra được chọn trong [None, 'numpy', 'torch', 'tensorflow', 'jax', 'arrow', 'pandas', 'polars'] . None means __getitem__ returns python objects (default).

- columns (List[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  None means __getitem__ returns all columns (default).
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
- **format_kwargs (additional keyword arguments) --
  Các đối số từ khóa được truyền cho hàm chuyển đổi như np.array , torch.tensor hoặc tensorflow.ragged.constant .0

Set __getitem__ return format (type and columns). The data formatting is applied on-the-bay.
The format type (for example "numpy") is used to format batches when using __getitem__ .

It's also possible to use custom transforms for formatting using with_transform().

Contrary to set_format(), with_format returns a new Dataset object.

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> ds = ds.map(lambda x: tokenizer(x['text'], truncation=True, padding=True), batched=True)
>>> ds.format
{'columns': ['text', 'label', 'input_ids', 'token_type_ids', 'attention_mask'],
 'format_kwargs': {},
 'output_all_columns': Sai,
 'type': None}
>>> ds = ds.with_format("torch")
>>> ds.format
{'columns': ['text', 'label', 'input_ids', 'token_type_ids', 'attention_mask'],
 'format_kwargs': {},
 'output_all_columns': Sai,
 'type': 'torch'}
>>> ds[0]
{'text': 'compassionately explores the seemingly irreconcilable situation between conservative chr
 'label': tensor(1),
 'input_ids': tensor([101, 18027, 16310, 16001,1103,9321,178, 11604,7235,6617,
          1742,2165,2820,1206,6588, 22572, 12937,1811,2153,1105,
          1147, 12890, 19587,6463,1105, 15026,1482,119,102,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0]),
 'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])}
```

with_transformdatasets.Dataset.with_transformhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2778[{"name": "transform", "val": ":

typing.Optional[typing.Callable]"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}]- transform (Callable , optional) -- User-defined formatting transform, replaces the format defined by set_format(). A formatting function is a callable that takes a batch (as a dict) as input and returns a batch. Hàm này được áp dụng ngay trước khi trả về các đối tượng trong __getitem__ .

- columns (List[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  Nếu được chỉ định thì lô đầu vào của phép biến đổi chỉ chứa các cột đó.
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
  Nếu được đặt thành True thì các cột chưa được định dạng khác sẽ được giữ nguyên với đầu ra của biến đổi.0
  Đặt định dạng trả về __getitem__ bằng cách sử dụng phép biến đổi này. Việc chuyển đổi được áp dụng n
  các đợt khi __getitem__ được gọi.

As set_format(), this can be reset using reset_format().

Contrary to set_transform(), with_transform returns a new Dataset object.

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> def encode(example):
...return tokenizer(example["text"], padding=True, truncation=True, return_tensors='pt')
>>> ds = ds.with_transform(encode)
>>> ds[0]
{'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1]),
  'input_ids': tensor([101, 18027, 16310, 16001,1103,9321,178, 11604,7235,6617,
               1742,2165,2820,1206,6588, 22572, 12937,1811,2153,1105,
               1147, 12890, 19587,6463,1105, 15026,1482,119,102]),
  'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
               0, 0, 0, 0, 0])}
```

getitemdatasets.Dataset.getitemhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2854[{"name": "key", "val": ""}]
Can be used to index columns (by string names) or rows (by integer index or iterable of indices or bools).

cleanup_cache_filesdatasets.Dataset.cleanup_cache_fileshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2867[] int Number of removed files.
Dọn dẹp tất cả các tệp bộ đệm trong thư mục bộ đệm của tập dữ liệu, ngoại trừ tệp bộ đệm hiện đang được có
một.

Hãy cẩn thận khi chạy lệnh này vì không có tiến trình nào khác hiện đang sử dụng bộ đệm khác tập tin.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds.cleanup_cache_files()
10
```

mapdatasets.Dataset.maphttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L2914[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "with_indices", "val": ": bool = False"}, {"name": "with_rank", "val": ": bool = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "drop_last_batch", "val": ": bool = False"}, {"name": "remove_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "cache_file_name", "val": ": typing.Optional[str] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "disable_nullable", "val": ": bool = False"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "suffix_template", "val": ": str = '_{rank:05d}of{num_proc:05d}'"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}, {"name": "desc", "val": ": typing.Optional[str] = None"}, {"name": "try_original_type",

"val": ": typing.Optional[bool] = True"}]- function (Callable) -- Function with one of the chữ ký sau:

- function(example: Dict[str, Any]) -> Dict[str, Any] if batched=False and with_indices=False and with_rank=False
- function(example: Dict[str, Any], *extra_args) -> Dict[str, Any] if batched=False and with_indices=True and/or with_rank=True (one extra arg for each)
- function(batch: Dict[str, List]) -> Dict[str, List] if batched=True and with_indices=False and with_rank=False
- function(batch: Dict[str, List], *extra_args) -> Dict[str, List] if batched=True and with_indices=True and/or with_rank=True (one extra arg for each)

Để sử dụng nâng cao, hàm cũng có thể trả về pyarrow.Table .
Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song.
Moreover if your function returns nothing (None), then map will run your function and return tập dữ liệu không thay đổi
Nếu không có chức năng nào được cung cấp, mặc định là chức năng nhận dạng: lambda x: x .

- with_indices (bool , defaults to False) -- Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này signature of function should be def function(example, idx[, rank]): ... .
- with_rank (bool , defaults to False) -- Cung cấp thứ hạng quy trình để hoạt động. Lưu ý rằng trong trường hợp này signature of function should be def function(example[, idx], rank): ... .
- input_columns (Optional[Union[str, List[str]]] , defaults to None) -- Các cột được truyền vào hàm như các đối số vị trí. Nếu Không có, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được chuyển dư một lý lẽ.
- batched (bool , defaults to False) -- Cung cấp hàng loạt ví dụ để hoạt động.
- batch_size (int , optional, defaults to 1000) -- Number of examples per batch provided to function if batched=True . If batch_size <= 0 or batch_size == None , provide the full dataset as a single batch to chức năng .
- drop_last_batch (bool , defaults to False) -- Liệu lô cuối cùng có nhỏ hơn batch_size hay không

bị loại bỏ thay vì được xử lý bởi hàm.

- remove_columns (Optional[Union[str, List[str]]] , defaults to None) --
  Xóa vùng chọn cột trong khi thực hiện ánh xạ.
  Các cột sẽ bị xóa trước khi cập nhật các ví dụ với đầu ra của hàm , tức là nếu
  chức năng đang thêm
  các cột có tên trong Remove_columns , các cột này sẽ được giữ lại.
- keep_in_memory (bool , defaults to False) --
  Giữ tập dữ liệu trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ tính toán hiện tại từ hàm
  có thể được xác định, hãy sử dụng nó thay vì tính toán lại.
- cache_file_name (str , optional, defaults to None) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  kết quả tính toán thay vì tên tệp bộ đệm được tạo tự động.
- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
  tốc độ xử lý.
  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn
  bộ nhớ tạm thời trong khi chạy bản đồ.
- features (Optional[datasets.Features] , defaults to None) --
  Sử dụng một Tính năng cụ thể để lưu trữ tệp bộ đệm
  thay vì cái được tạo tự động.
- disable_nullable (bool , defaults to False) --
  Không cho phép giá trị null trong bảng.
- fn_kwargs (Dict , optional, defaults to None) --
  Đối số từ khóa được truyền cho hàm.
- num_proc (int , optional, defaults to None) --
  Số lượng tiến trình được sử dụng cho đa xử lý.
  Nếu Không có hoặc 0 , thì không có đa xử lý nào được sử dụng và thao tác sẽ chạy trong quy trình chí
  Nếu lớn hơn 1 , một hoặc nhiều quy trình công nhân được sử dụng để xử lý dữ liệu trong
  song song.
  Note: The function passed to map() must be picklable for multiprocessing to work
  chính xác
  (i.e., prefer functions defined at the top level of a module, not inside another function

or class).
suffix_template (str):
Nếu cache_file_name được chỉ định thì hậu tố này
sẽ được thêm vào cuối tên cơ sở của mỗi tên. Mặc định là
"_{rank:05d}_of_{num_proc:05d}" . For example, if cache_file_name is
"processed.arrow", sau đó cho
rank=1 and num_proc=4 , the resulting file would be
"processed_00001_of_00004.arrow" cho hậu tố mặc định.

- new_fingerprint (str , optional, defaults to None) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi các lập luận.
- desc (str , optional, defaults to None) --
  Mô tả có ý nghĩa sẽ được hiển thị cùng với thanh tiến trình trong khi lập bản đồ
  ví dụ.
- try_original_type (Optional[bool] , defaults to True) --
  Try to keep the types of the original columns (e.g. int32 -> int32).
  Đặt thành Sai nếu bạn muốn luôn suy ra các loại mới.0

Apply a function to all the examples in the table (individually or in batches) and update the
bàn.
Nếu hàm của bạn trả về một cột đã tồn tại thì nó sẽ ghi đè lên cột đó.

Bạn có thể chỉ định xem hàm có nên được bó hay không bằng tham số được bó:

- Nếu bó là Sai thì hàm sẽ lấy 1 mẫu và trả về 1 mẫu.
  An example is a dictionary, e.g. {"text": "Hello there !"} .
- Nếu batched là True và batch_size là 1 thì hàm sẽ lấy một batch gồm 1 ví dụ như
  đầu vào và có thể trả về một lô có 1 hoặc nhiều mẫu.
  A batch is a dictionary, e.g. a batch of 1 example is {"text": ["Hello there !"]} .
- If batched is True and batch_size is n > 1 , then the function takes a batch of n
  ví dụ làm đầu vào và có thể trả về một lô có n ví dụ hoặc với số lượng tùy ý
  ví dụ.
  Lưu ý rằng đợt cuối cùng có thể có ít hơn n mẫu.
  A batch is a dictionary, e.g. a batch of n examples is {"text": ["Hello there !"] * n} .

Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song, với tối đa một nghìn cuộc gọi đồng thời.
Bạn nên sử dụng asyncio.Semaphore trong hàm của mình nếu bạn muốn đặt mức tối đa số thao tác có thể thực hiện đồng thời.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> def add_prefix(example):
...example["text"] = "Review: " + example["text"]
...trả lại ví dụ
>>> ds = ds.map(add_prefix)
>>> ds[0:3]["text"]
['Review: compassionately explores the seemingly irreconcilable situation between conservative chr
 'Đánh giá: chỉ riêng phần nhạc phim đã đáng giá vào cửa .',
 'Đánh giá: Rodriguez đã thực hiện một công việc tuyệt vời trong việc phân tích chủng tộc theo phong cách

# xử lý một loạt ví dụ
>>> ds = ds.map(lambda example: tokenizer(example["text"]), batched=True)
# đặt số lượng bộ xử lý
>>> ds = ds.map(add_prefix, num_proc=4)
```

filterdatasets.Dataset.filterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L3792[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "with_indices", "val": ": bool = False"}, {"name": "with_rank", "val": ": bool = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "cache_file_name", "val": ": typing.Optional[str] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "suffix_template", "val": ": str = '_{rank:05d}of{num_proc:05d}'"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}, {"name": "desc", "val": ": typing.Optional[str] = None"}]- function (Callable) --
Có thể gọi được với một trong các chữ ký sau:

- function(example: Dict[str, Any]) -> bool if batched=False and with_indices=False and with_rank=False
- function(example: Dict[str, Any], *extra_args) -> bool if batched=False and with_indices=True and/or with_rank=True (one extra arg for each)
- function(batch: Dict[str, List]) -> List[bool] if batched=True and with_indices=False and with_rank=False
- function(batch: Dict[str, List], *extra_args) -> List[bool] if batched=True and with_indices=True and/or with_rank=True (one extra arg for each)

Nếu hàm không đồng bộ thì bộ lọc sẽ chạy song song hàm của bạn.
Nếu không có hàm nào được cung cấp, mặc định là hàm luôn True: lambda x: True .

- with_indices (bool , defaults to False) --
  Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này signature of function should be def function(example, idx[, rank]): ... .
- with_rank (bool , defaults to False) --
  Cung cấp thứ hạng quy trình để hoạt động. Lưu ý rằng trong trường hợp này signature of function should be def function(example[, idx], rank): ... .
- input_columns (str or List[str] , optional) --
  Các cột được chuyển vào chức năng như những lập luận mang tính vị trí. Nếu Không, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được ch lý lẽ.
- batched (bool , defaults to False) --
  Cung cấp hàng loạt ví dụ để hoạt động.
- batch_size (int , optional, defaults to 1000) --
  Số mẫu mỗi lô được cung cấp để hoạt động nếu batched = True . If batched = False , one example per batch is passed to function . If batch_size <= 0 or batch_size == None , provide the full dataset as a single batch to chức năng .
- keep_in_memory (bool , defaults to False) --
  Giữ tập dữ liệu trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ tính toán hiện tại từ hàm có thể được xác định, hãy sử dụng nó thay vì tính toán lại.
- cache_file_name (str , optional) --

Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
kết quả tính toán thay vì tên tệp bộ đệm được tạo tự động.

• writer_batch_size (int , defaults to 1000) --

Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
tốc độ xử lý.
Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn
bộ nhớ tạm thời trong khi chạy bản đồ.

• fn_kwargs (dict , optional) --

Đối số từ khóa được truyền cho hàm.

• num_proc (int , optional, defaults to None) --

Số lượng tiến trình được sử dụng cho đa xử lý.
Nếu Không có hoặc 0 , thì không có đa xử lý nào được sử dụng và thao tác sẽ chạy trong quy trình chí
Nếu lớn hơn 1 , một hoặc nhiều quy trình công nhân được sử dụng để xử lý dữ liệu trong
song song.
Note: The function passed to map() must be picklable for multiprocessing to work
chính xác
(i.e., prefer functions defined at the top level of a module, not inside another function
or class).

• suffix_template (str) --

Nếu cache_file_name được chỉ định thì hậu tố này sẽ được thêm vào cuối tên cơ sở
của mỗi người.
For example, if cache_file_name is "processed.arrow" , then for rank = 1 and
num_proc = 4 ,
the resulting file would be "processed_00001_of_00004.arrow" for the default suffix (default
_{rank:05d}_of_{num_proc:05d}).

• new_fingerprint (str , optional) --

Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và
biến đổi các lập luận.

• desc (str , optional, defaults to None) --

Mô tả có ý nghĩa sẽ được hiển thị cùng với thanh tiến trình trong khi lọc
ví dụ.0
Áp dụng chức năng lọc cho tất cả các phần tử trong bảng theo đợt
và cập nhật bảng để tập dữ liệu chỉ bao gồm các ví dụ theo bộ lọc

chức năng.

Nếu hàm không đồng bộ thì bộ lọc sẽ chạy hàm của bạn song song, với tối đa một thousand simultaneous calls (configurable).
Bạn nên sử dụng asyncio.Semaphore trong hàm của mình nếu bạn muốn đặt mức tối đa số thao tác có thể thực hiện đồng thời.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds = ds.filter(lambda x: x["label"] == 1)
>>> ds
Dataset({
    features: ['text', 'label'],
    số_hàng: 533
})
```

selectdatasets.Dataset.selecthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4019[{"name": "indices", "val": ": Iterable"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "indices_cache_file_name", "val": ": typing.Optional[str] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]- indices (range , list , iterable , ndarray or Series) --
Phạm vi, danh sách hoặc mảng 1D của các chỉ số nguyên để lập chỉ mục.
Nếu các chỉ số tương ứng với một phạm vi liền kề, bảng Mũi tên sẽ được cắt đơn giản.
Tuy nhiên, việc chuyển một danh sách các chỉ mục không liền kề sẽ tạo ra ánh xạ chỉ mục, đó là kém hiệu quả hơn nhiều,
nhưng vẫn nhanh hơn việc tạo lại bảng Mũi tên được tạo từ các hàng được yêu cầu.

- keep_in_memory (bool , defaults to False) --
  Giữ ánh xạ chỉ mục trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- indices_cache_file_name (str , optional, defaults to None) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các ánh xạ chỉ mục thay vì tên tệp bộ đệm được tạo tự động.
- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.

Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và tốc độ xử lý.
Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn bộ nhớ tạm thời trong khi chạy bản đồ.

- new_fingerprint (str , optional, defaults to None) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số.0
  Tạo tập dữ liệu mới với các hàng được chọn theo danh sách/mảng chỉ mục.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds = ds.select(range(4))
>>> ds
Dataset({
    features: ['text', 'label'],
    số_hàng: 4
})
```

Sortdatasets.Dataset.sorthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4357[{"name": "column_names", "val": ": typing.Union[str, collections.abc.Sequence[str]]"}, {"name": "reverse", "val": ": typing.Union[bool, collections.abc.Sequence[bool]] = False"}, {"name": "null_placement", "val": ": str = 'at_end'"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "indices_cache_file_name", "val": ": typing.Optional[str] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]- column_names (Union[str, Sequence[str]]) --
Column name(s) to sort by.

- reverse (Union[bool, Sequence[bool]] , defaults to False) --
  Nếu Đúng, hãy sắp xếp theo thứ tự giảm dần thay vì tăng dần. Nếu một bool duy nhất được cung cấp, giá trị được áp dụng để sắp xếp tất cả các tên cột. Nếu không thì danh sách các bools có
  Phải cung cấp cùng độ dài và thứ tự như tên cột.
- null_placement (str , defaults to at_end) --

Đặt giá trị Không có ở đầu nếu at_start hoặc đầu tiên hoặc ở cuối nếu at_end hoặc cuối cùng

- keep_in_memory (bool , defaults to False) --
  Giữ các chỉ mục đã sắp xếp trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ các chỉ mục được sắp xếp
  có thể được xác định, hãy sử dụng nó thay vì tính toán lại.
- indices_cache_file_name (str , optional, defaults to None) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  các chỉ mục được sắp xếp thay vì tên tệp bộ đệm được tạo tự động.
- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị cao hơn mang lại tệp bộ đệm nhỏ hơn, giá trị thấp hơn sẽ tiêu tốn ít bộ nhớ tạm thời hơn.
- new_fingerprint (str , optional, defaults to None) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số0
  Tạo tập dữ liệu mới được sắp xếp theo một hoặc nhiều cột.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset('cornell-movie-review-data/rotten_tomatoes', split='validation')
>>> ds['label'][:10]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
>>> sorted_ds = ds.sort('label')
>>> sorted_ds['label'][:10]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
>>> another_sorted_ds = ds.sort(['label', 'text'], reverse=[True, False])
>>> another_sorted_ds['label'][:10]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

shuffledatasets.Dataset.shufflehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4485[{"name": "seed", "val": ": typing.Optional[int] = None"}, {"name": "generator", "val": ": typing.Optional[numpy.random._generator.Generator] = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "indices_cache_file_name", "val": ":

typing.Optional[str] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "new_fingerprint", "val": ": typing.Optional[str] = None"}]- seed (int , optional) None
A seed to initialize the default BitGenerator if generator=None .
Nếu Không có thì entropy mới, không thể đoán trước sẽ được lấy khỏi HĐH.
If an int or array_like[ints] is passed, then it will be passed to SeedSequence to derive trạng thái BitGenerator ban đầu.

- generator (numpy.random.Generator , optional) --
  Trình tạo ngẫu nhiên Numpy được sử dụng để tính toán hoán vị của các hàng tập dữ liệu.
  If generator=None (default), uses np.random.default_rng (the default BitGenerator (PCG64) of NumPy).
- keep_in_memory (bool , default False) --
  Giữ các chỉ mục được xáo trộn trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ các chỉ mục được xáo trộn
  có thể được xác định, hãy sử dụng nó thay vì tính toán lại.
- indices_cache_file_name (str , optional) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  các chỉ mục được xáo trộn thay vì tên tệp bộ đệm được tạo tự động.
- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
  tốc độ xử lý.
  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn
  bộ nhớ tạm thời trong khi chạy bản đồ.
- new_fingerprint (str , optional, defaults to None) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số.0
  Tạo Bộ dữ liệu mới trong đó các hàng được xáo trộn.

Hiện tại việc xáo trộn sử dụng các trình tạo ngẫu nhiên khó hiểu.
Bạn có thể cung cấp NumPy BitGenerator để sử dụng hoặc hạt giống để bắt đầu mặc định của NumPy random generator (PCG64).

Shuffling takes the list of indices [0:len(my_dataset)] and shuffles it to create an indices
lập bản đồ.
Tuy nhiên, ngay sau khi Tập dữ liệu của bạn có ánh xạ chỉ mục, tốc độ có thể chậm hơn gấp 10 lần.
Điều này là do có thêm một bước để đọc chỉ mục hàng bằng cách sử dụng ánh xạ chỉ mục,
và quan trọng nhất, bạn không còn đọc những khối dữ liệu liền kề nhau nữa.
Để khôi phục tốc độ, bạn cần phải ghi lại toàn bộ tập dữ liệu trên đĩa của mình bằng cách sử dụng
Dataset.flatten_indices(), which removes the indices mapping.

Tuy nhiên, việc này có thể mất nhiều thời gian tùy thuộc vào kích thước tập dữ liệu của bạn:

```
my_dataset[0]# fast
my_dataset = my_dataset.shuffle(seed=42)
my_dataset[0]# up to 10x slower
my_dataset = my_dataset.flatten_indices()# rewrite the shuffled dataset on disk as contiguous ch
my_dataset[0]# fast again
```

Trong trường hợp này, chúng tôi khuyên bạn nên chuyển sang IterableDataset và tận dụng tốc độ nhanh củ
approximate shuffling method IterableDataset.shuffle().

Nó chỉ xáo trộn thứ tự phân đoạn và thêm bộ đệm ngẫu nhiên vào tập dữ liệu của bạn, giúp giữ nguyên
tốc độ tối ưu của tập dữ liệu của bạn:

```
my_iterable_dataset = my_dataset.to_iterable_dataset(num_shards=128)
for example in enumerate(my_iterable_dataset):# fast
    vượt qua

shuffled_iterable_dataset = my_iterable_dataset.shuffle(seed=42, buffer_size=100)

for example in enumerate(shuffled_iterable_dataset):# as fast as before
    vượt qua
```

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds['label'][:10]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

# đặt hạt giống
>>> shuffled_ds = ds.shuffle(seed=42)
>>> shuffled_ds['label'][:10]
[1, 0, 1, 1, 0, 0, 0, 0, 0, 0]
```

Skipdatasets.Dataset.skiphttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4272[{"name": "n", "val": ": int"}]- n (int) --
Số phần tử cần bỏ qua.0

Tạo Bộ dữ liệu mới bỏ qua n phần tử đầu tiên.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> list(ds.take(3))
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'}]
>>> ds = ds.skip(1)
>>> list(ds.take(3))
[{'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'},
  {'label': 1,
  'text': 'nếu thỉnh thoảng bạn thích đi xem phim để giải trí , wasabi là nơi tốt để bắt đầu
```

takedatasets.Dataset.takehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4334[{"name": "n", "val": ": int"}]- n (int) --
Số phần tử cần lấy.0

Tạo một Bộ dữ liệu mới chỉ có n phần tử đầu tiên.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> small_ds = ds.take(2)
>>> list(small_ds)
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
```

train_test_splitdatasets.Dataset.train_test_splithttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4617[{"name": "test_size", "val": ": typing.Union[float, int, NoneType] = None"}, {"name": "train_size", "val": ": typing.Union[float, int, NoneType] = None"}, {"name": "shuffle", "val": ": bool = True"}, {"name": "stratify_by_column", "val": ": typing.Optional[str] = None"}, {"name": "seed", "val": ": typing.Optional[int] = None"}, {"name": "generator", "val": ": typing.Optional[numpy.random._generator.Generator] = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "train_indices_cache_file_name", "val": ": typing.Optional[str] = None"}, {"name": "test_indices_cache_file_name", "val": ": typing.Optional[str] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "train_new_fingerprint", "val": ": typing.Optional[str] = None"}, {"name": "test_new_fingerprint", "val": ": typing.Optional[str] = None"}]- test_size (Union[float, int, None] , optional) --
Kích thước của phần chia thử nghiệm
Nếu float , phải nằm trong khoảng từ 0,0 đến 1,0 và biểu thị tỷ lệ của tập dữ liệu với đưa vào phần phân chia thử nghiệm.
Nếu int , biểu thị số lượng mẫu thử tuyệt đối.
Nếu Không có, giá trị được đặt thành phần bù của kích thước tàu.
Nếu train_size cũng là None , nó sẽ được đặt thành 0,25 .

- train_size (Union[float, int, None] , optional) --
  Kích thước của tàu chia
  Nếu float , phải nằm trong khoảng từ 0,0 đến 1,0 và thể hiện tỷ lệ của tập dữ liệu với bao gồm trong việc chia tàu.

Nếu int , đại diện cho số lượng mẫu tàu tuyệt đối.

Nếu Không có, giá trị sẽ tự động được đặt thành phần bù của kích thước kiểm tra.

- shuffle (bool , optional, defaults to True) --

  Có nên xáo trộn dữ liệu trước khi chia tách hay không.

- stratify_by_column (str , optional, defaults to None) --

  Tên cột của nhãn được sử dụng để thực hiện phân chia dữ liệu theo tầng.

- seed (int , optional) --

  A seed to initialize the default BitGenerator if generator=None .

  Nếu Không có thì entropy mới, không thể đoán trước sẽ được lấy khỏi HĐH.

  If an int or array_like[ints] is passed, then it will be passed to SeedSequence to

  lấy được trạng thái BitGenerator ban đầu.

- generator (numpy.random.Generator , optional) --

  Trình tạo ngẫu nhiên Numpy được sử dụng để tính toán hoán vị của các hàng tập dữ liệu.

  If generator=None (default), uses np.random.default_rng (the default BitGenerator

  (PCG64) of NumPy).

- keep_in_memory (bool , defaults to False) --

  Giữ các chỉ mục phân chia trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.

- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --

  Nếu một tệp bộ đệm lưu trữ các chỉ mục phân tách

  có thể được xác định, hãy sử dụng nó thay vì tính toán lại.

- train_cache_file_name (str , optional) --

  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các

  đào tạo các chỉ số phân chia thay vì tên tệp bộ đệm được tạo tự động.

- test_cache_file_name (str , optional) --

  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các

  kiểm tra các chỉ mục phân tách thay vì tên tệp bộ đệm được tạo tự động.

- writer_batch_size (int , defaults to 1000) --

  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.

  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và

  tốc độ xử lý.

  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn

  bộ nhớ tạm thời trong khi chạy bản đồ.

- train_new_fingerprint (str , optional, defaults to None) --

  Dấu vân tay mới của đoàn tàu sau khi biến hình.

  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và

biến đổi đối số

- test_new_fingerprint (str , optional, defaults to None) --
  Dấu vân tay mới của bộ kiểm tra sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và biến đổi đối số0
  Return a dictionary (datasets.DatasetDict) with two random train and test subsets (train and test Dataset splits).
  Việc phân tách được tạo từ tập dữ liệu theo test_size , train_size và shuffle .

Phương pháp này tương tự như scikit-learn train_test_split.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds = ds.train_test_split(test_size=0.2, shuffle=True)
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 852
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 214
    })
})

# đặt hạt giống
>>> ds = ds.train_test_split(test_size=0.2, seed=42)

# phân tầng
>>> ds = load_dataset("imdb",split="train")
Dataset({
    features: ['text', 'label'],
    num_rows: 25000
})
>>> ds = ds.train_test_split(test_size=0.2, stratify_by_column="label")
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        số_hàng: 20000
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 5000
    })
})
```

sharddatasets.Dataset.shardhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4900[{"name": "num_shards", "val": ": int"}, {"name": "index", "val": ": int"}, {"name": "contiguous", "val": ": bool = True"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "indices_cache_file_name", "val": ": typing.Optional[str] = None"}, {"name":

"writer_batch_size", "val": ": typing.Optional[int] = 1000"}]- num_shards (int) --
Có bao nhiêu phân đoạn để chia tập dữ liệu thành.

- index (int) --
  Phân đoạn nào để chọn và trả lại.
- contiguous -- (bool , defaults to True):
  Có nên chọn các khối chỉ mục liền kề cho phân đoạn hay không.
- keep_in_memory (bool , defaults to False) --
  Giữ tập dữ liệu trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- indices_cache_file_name (str , optional) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  chỉ mục của từng phân đoạn thay vì tên tệp bộ đệm được tạo tự động.
- writer_batch_size (int , defaults to 1000) --
  Điều này chỉ liên quan đến ánh xạ chỉ số.
  Số lượng chỉ mục cho mỗi thao tác ghi đối với trình ghi tệp bộ nhớ đệm.
  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
  tốc độ xử lý.
  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn
  bộ nhớ tạm thời trong khi chạy bản đồ .0
  Trả về phân đoạn chỉ mục -nth từ tập dữ liệu được chia thành các phần num_shards.

This shards deterministically. dataset.shard(n, i) splits the dataset into contiguous chunks,
so it can be easily concatenated back together after processing. If len(dataset) % n == l ,
sau đó
first l dataset each have length (len(dataset) // n) + 1 , and the remaining dataset have
length (len(dataset) // n) .
datasets.concatenate_datasets([dset.shard(n, i) for i in range(n)]) returns a dataset with
thứ tự giống như bản gốc.

Note: n should be less or equal to the number of elements in the dataset len(dataset) .

On the other hand, dataset.shard(n, i, contiguous=False) contains all elements of the
dataset whose index mod n = i .

Be sure to shard before using any randomizing operator (such as shuffle).
Tốt nhất là nên sử dụng toán tử phân đoạn sớm trong quy trình dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation")
>>> ds
Dataset({
    features: ['text', 'label'],
    num_rows: 1066
})
>>> ds = ds.shard(num_shards=2, index=0)
>>> ds
Dataset({
    features: ['text', 'label'],
    số_hàng: 533
})
```

lặp lạidatasets.Dataset.repeathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4302[{"name": "num_times", "val": ": int"}]- num_times (int) --
Số lần lặp lại tập dữ liệu.0

Tạo Tập dữ liệu mới lặp lại tập dữ liệu cơ bản num_times.

Giống như itertools.repeat, việc lặp lại một lần sẽ trả về toàn bộ tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> ds = ds.take(2).repeat(2)
>>> list(ds)
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'},
  {'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'}]
```

to_tf_datasetdatasets.Dataset.to_tf_datasethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L331[{"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "shuffle", "val": ": bool = False"}, {"name": "collate_fn", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "drop_remainder", "val": ": bool = False"}, {"name": "collate_fn_args", "val": ": typing.Optional[dict[str, typing.Any]] = None"}, {"name": "label_cols", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "prefetch", "val": ": bool = True"}, {"name": "num_workers", "val": ": int = 0"}, {"name": "num_test_batches", "val": ": int = 20"}]- batch_size (int , optional) --
Kích thước của các lô cần tải từ tập dữ liệu. Mặc định là Không có, nghĩa là tập dữ liệu sẽ không là
batched, but the returned dataset can be batched later with tf_dataset.batch(batch_size) .

- columns (List[str] or str , optional) --
  Dataset column(s) to load in the tf.data.Dataset .
  Tên cột được tạo bởi collate_fn và không tồn tại trong bản gốc
  tập dữ liệu có thể được sử dụng.
- shuffle(bool , defaults to False) --
  Xáo trộn thứ tự tập dữ liệu khi tải. Đề xuất Đúng cho đào tạo, Sai cho
  xác nhận/đánh giá.
- drop_remainder(bool , defaults to False) --
  Bỏ lô chưa hoàn thiện cuối cùng khi tải. Đảm bảo

rằng tất cả các lô do tập dữ liệu mang lại sẽ có cùng độ dài trên thứ nguyên lô.

- collate_fn(Callable , optional) --
  A function or callable object (such as a DataCollator) that will collate
  danh sách các mẫu thành một lô.
- collate_fn_args (Dict , optional) --
  Một mệnh lệnh tùy chọn về các đối số từ khóa sẽ được chuyển đến
  đối chiếu_fn .
- label_cols (List[str] or str , defaults to None) --
  Dataset column(s) to load as labels.
  Lưu ý rằng nhiều mô hình tính toán tổn thất nội bộ thay vì để Keras thực hiện việc đó, trong đó trường hợp
  chuyển nhãn ở đây là tùy chọn, miễn là chúng nằm trong cột đầu vào .
- prefetch (bool , defaults to True) --
  Có chạy trình tải dữ liệu trong một luồng riêng biệt và duy trì hay không
  một bộ đệm nhỏ các đợt để đào tạo. Cải thiện hiệu suất bằng cách cho phép tải dữ liệu trong
  nền trong khi mô hình đang được huấn luyện.
- num_workers (int , defaults to 0) --
  Số lượng công nhân được sử dụng để tải tập dữ liệu.
- num_test_batches (int , defaults to 20) --
  Số lô được sử dụng để suy ra chữ ký đầu ra của tập dữ liệu.
  Con số này càng cao thì chữ ký sẽ càng chính xác nhưng sẽ mất nhiều thời gian hơn.
  ĐẾN
  tạo tập dữ liệu.0 tf.data.Dataset
  Tạo một tf.data.Dataset từ Bộ dữ liệu cơ bản. tf.data.Dataset này sẽ tải
  và đối chiếu các lô từ
  the Dataset, and is suitable for passing to methods like model.fit() or model.predict() .
  Tập dữ liệu sẽ mang lại
  dict cho cả đầu vào và nhãn trừ khi dict chỉ chứa một khóa duy nhất, trong
  trường hợp nào là thô
  thay vào đó, tf.Tensor được mang lại.

Ví dụ:

```
>>> ds_train = ds["train"].to_tf_dataset(
...columns=['input_ids', 'token_type_ids', 'attention_mask', 'label'],
...shuffle=True,
...batch_size=16,
...collate_fn=data_collator,
... )
```

push_to_hubdatasets.Dataset.push_to_hubhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L5641[{"name": "repo_id", "val": ": str"}, {"name": "config_name", "val": ": str = 'default'"}, {"name": "set_default", "val": ": typing.Optional[bool] = None"}, {"name": "split", "val": ": typing.Optional[str] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "commit_message", "val": ": typing.Optional[str] = None"}, {"name": "commit_description", "val": ": typing.Optional[str] = None"}, {"name": "private", "val": ": typing.Optional[bool] = None"}, {"name": "token", "val": ": typing.Optional[str] = None"}, {"name": "revision", "val": ": typing.Optional[str] = None"}, {"name": "create_pr", "val": ": typing.Optional[bool] = False"}, {"name": "max_shard_size", "val": ": typing.Union[str, int, NoneType] = None"}, {"name": "num_shards", "val": ": typing.Optional[int] = None"}, {"name": "embed_external_files", "val": ": bool = True"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}]- repo_id (str) --
The ID of the repository to push to in the following format: <user>/<dataset_name> or <org>/<dataset_name> . Also accepts <dataset_name> , which will default to the namespace của người dùng đã đăng nhập.

- config_name (str , defaults to "default") --
  The configuration name (or subset) of a dataset. Defaults to "default".
- set_default (bool , optional) --
  Có đặt cấu hình này làm cấu hình mặc định hay không. Ngược lại, cấu hình mặc định là cái một
  được đặt tên là "mặc định".
- split (str , optional) --
  Tên của phần tách sẽ được đặt cho tập dữ liệu đó. Mặc định là self.split .
- data_dir (str , optional) --
  Tên thư mục sẽ chứa các tệp dữ liệu được tải lên. Mặc định là config_name nếu khác biệt
  từ "mặc định", "dữ liệu" khác.

• commit_message (str , optional) --

Thông báo để cam kết trong khi đẩy. Sẽ mặc định là "Tải lên tập dữ liệu".

• commit_description (str , optional) --

Mô tả cam kết sẽ được tạo.

Additionally, description of the PR if a PR is created (create_pr is True).

• private (bool , optional) --

Whether to make the repo private. If None (default), the repo will be public unless the

mặc định của tổ chức là riêng tư. Giá trị này bị bỏ qua nếu repo đã tồn tại.

• token (str , optional) --

Mã thông báo xác thực tùy chọn cho Hugging Face Hub. Nếu không có mã thông báo nào được chuyển, mặc định

vào mã thông báo được lưu cục bộ khi đăng nhập bằng deathface-cli login . Sẽ gây ra lỗi

nếu không có mã thông báo nào được chuyển và người dùng chưa đăng nhập.

• revision (str , optional) --

Branch để đẩy các tập tin đã tải lên. Mặc định là nhánh "chính".

• create_pr (bool , optional, defaults to False) --

Có nên tạo PR bằng các tệp đã tải lên hay cam kết trực tiếp.

• max_shard_size (int or str , optional, defaults to "500MB") --

Kích thước tối đa của các phân đoạn dữ liệu sẽ được tải lên trung tâm. Nếu được biểu thị dưới dạng chuỗi, cần phải là chữ số theo sau là

a unit (like "5MB").

• num_shards (int , optional) --

Số lượng mảnh để viết. Theo mặc định, số lượng phân đoạn phụ thuộc vào max_shard_size .

• embed_external_files (bool , defaults to True) --

Có nhúng byte tệp vào phân đoạn hay không.

Cụ thể, điều này sẽ thực hiện những việc sau trước khi đẩy các trường thuộc loại:

Âm thanh và Hình ảnh: xóa thông tin đường dẫn cục bộ và nhúng nội dung tệp vào

Tập tin sàn gỗ.

• num_proc (int , optional, defaults to None) --

Số lượng quy trình khi chuẩn bị và tải lên tập dữ liệu.

Điều này hữu ích nếu tập dữ liệu được tạo từ nhiều mẫu hoặc tệp phương tiện để nhúng.

Đa xử lý bị tắt theo mặc định.

0huggingface_hub.CommitInfo

Đẩy tập dữ liệu vào trung tâm dưới dạng tập dữ liệu Parquet.

Tập dữ liệu được đẩy bằng các yêu cầu HTTP và không cần phải có git hoặc git-

lfs được cài đặt.

Theo mặc định, các tệp Parquet kết quả được tự chứa. Nếu tập dữ liệu của bạn chứa Hình ảnh, Âm thanh hoặc Video
dữ liệu, tệp Parquet sẽ lưu trữ byte hình ảnh hoặc tệp âm thanh của bạn.
Bạn có thể tắt tính năng này bằng cách đặt embed_external_files thành False .

Ví dụ:

```
>>> dataset.push_to_hub("<organization>/<dataset_id>")
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>", private=True)
>>> dataset.push_to_hub("<organization>/<dataset_id>", max_shard_size="1GB")
>>> dataset.push_to_hub("<organization>/<dataset_id>", num_shards=1024)
```

If your dataset has multiple splits (e.g. train/validation/test):

```
>>> train_dataset.push_to_hub("<organization>/<dataset_id>", split="train")
>>> val_dataset.push_to_hub("<organization>/<dataset_id>", split="validation")
>>> # later
>>> dataset = load_dataset("<organization>/<dataset_id>")
>>> train_dataset = dataset["train"]
>>> val_dataset = dataset["validation"]
```

If you want to add a new configuration (or subset) to a dataset (e.g. if the dataset has multiple tasks/versions/languages):

```
>>> english_dataset.push_to_hub("<organization>/<dataset_id>", "en")
>>> french_dataset.push_to_hub("<organization>/<dataset_id>", "fr")
>>> # later
>>> english_dataset = load_dataset("<organization>/<dataset_id>", "en")
>>> french_dataset = load_dataset("<organization>/<dataset_id>", "fr")
```

save_to_diskdatasets.Dataset.save_to_diskhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1493[{"name": "dataset_path", "val": ": typing.Union[str, bytes, os.PathLike]"}, {"name": "max_shard_size", "val": ": typing.Union[str, int, NoneType] = None"}, {"name": "num_shards", "val": ": typing.Optional[int] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict]

= None"}]- dataset_path (path-like) --
Path (e.g. dataset/train) or remote URI (e.g. s3://my-bucket/dataset/train)
của thư mục tập dữ liệu nơi tập dữ liệu sẽ được lưu vào.

- max_shard_size (int or str , optional, defaults to "500MB") --
  Kích thước tối đa của các phân đoạn dữ liệu sẽ được lưu vào hệ thống tệp. Nếu được biểu thị dưới dạng chuỗi, cần phải là chữ số theo sau là đơn vị
  (like "50MB").
- num_shards (int , optional) --
  Số lượng mảnh để viết. Theo mặc định, số lượng phân đoạn phụ thuộc vào max_shard_size và num_proc.
- num_proc (int , optional) --
  Số lượng quy trình khi tải xuống và tạo tập dữ liệu cục bộ.
  Đa xử lý bị tắt theo mặc định.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0

Lưu tập dữ liệu vào thư mục tập dữ liệu hoặc trong hệ thống tệp bằng cách sử dụng bất kỳ triển khai nào củ
fsspec.spec.AbstractFileSystem .

Đối với dữ liệu Hình ảnh, Âm thanh và Video:

All the Image(), Audio() and Video() data are stored in the arrow files.
If you want to store paths or urls, please use the Value("string") type.

Ví dụ:

```
>>> ds.save_to_disk("path/to/dataset/directory")
>>> ds.save_to_disk("path/to/dataset/directory", max_shard_size="1GB")
>>> ds.save_to_disk("path/to/dataset/directory", num_shards=1024)
```

Load_from_diskdatasets.Dataset.load_from_diskhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1687[{"name": "dataset_path", "val": ": typing.Union[str, bytes, os.PathLike]"}, {"name": "keep_in_memory", "val": ": typing.Optional[bool] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}]- dataset_path (path-like)
None

Path (e.g. "dataset/train") or remote URI (e.g. "s3//my-bucket/dataset/train")
của thư mục tập dữ liệu nơi tập dữ liệu sẽ được tải từ đó.

- keep_in_memory (bool , defaults to None) --
  Có sao chép tập dữ liệu vào bộ nhớ hay không. Nếu Không , thì
  tập dữ liệu sẽ không được sao chép trong bộ nhớ trừ khi được bật rõ ràng bằng cách cài đặt
  bộ dữ liệu.config.IN_MEMORY_MAX_SIZE thành khác không. Xem thêm chi tiết tại
  phần cải thiện hiệu suất.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0Dataset hoặc DatasetDict- Nếu tập dữ liệu_path là đường dẫn của thư mục tập dữ liệu, tập dữ liệu
  được yêu cầu.
- Nếu tập dữ liệu_path là đường dẫn của thư mục tập dữ liệu dict, thì một tập dữ liệu.DatasetDict với mỗi t
  tách ra.

Tải tập dữ liệu đã được lưu trước đó bằng save_to_disk từ thư mục tập dữ liệu hoặc
từ một
hệ thống tập tin bằng cách sử dụng bất kỳ triển khai fsspec.spec.AbstractFileSystem nào.

Ví dụ:

```
>>> ds = load_from_disk("path/to/dataset/directory")
```

Flatten_indicesdatasets.Dataset.flatten_indiceshttps://github.com/huggingface/datasets/blob/
4.2.0/src/datasets/arrow_dataset.py#L3940[{"name": "keep_in_memory", "val": ": bool =
False"}, {"name": "cache_file_name", "val": ": typing.Optional[str] = None"}, {"name":
"writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "features", "val": ":
typing.Optional[datasets.features.features.Features] = None"}, {"name": "disable_nullable",
"val": ": bool = False"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name":
"new_fingerprint", "val": ": typing.Optional[str] = None"}]- keep_in_memory (bool , defaults to
False) --
Giữ tập dữ liệu trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.

- cache_file_name (str , optional, default None) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  kết quả tính toán thay vì tên tệp bộ đệm được tạo tự động.

- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
  tốc độ xử lý.
  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn
  bộ nhớ tạm thời trong khi chạy bản đồ.
- features (Optional[datasets.Features] , defaults to None) --
  Sử dụng một Tính năng cụ thể để lưu trữ tệp bộ đệm
  thay vì cái được tạo tự động.
- disable_nullable (bool , defaults to False) --
  Cho phép giá trị null trong bảng.
- num_proc (int , optional, default None) --
  Số lượng quy trình tối đa khi tạo bộ đệm. Các phân đoạn đã được lưu trong bộ nhớ đệm đã được tải
  tuần tự
- new_fingerprint (str , optional, defaults to None) --
  Dấu vân tay mới của tập dữ liệu sau khi biến đổi.
  Nếu Không , dấu vân tay mới được tính bằng cách sử dụng hàm băm của dấu vân tay trước đó và
  biến đổi đối số0
  Tạo và lưu trữ Bộ dữ liệu mới bằng cách làm phẳng ánh xạ chỉ mục.

to_csvdatasets.Dataset.to_csvhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L4977[{"name": "path_or_buf", "val": ": typing.Union[str, bytes, os.PathLike, typing.BinaryIO]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**to_csv_kwargs", "val": ""}]- path_or_buf (PathLike or FileOrBuffer) --
Either a path to a file (e.g. file.csv), a remote URI (e.g.
hf://datasets/username/my_dataset_name/data.csv),
hoặc BinaryIO, nơi tập dữ liệu sẽ được lưu ở định dạng đã chỉ định.

- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .
- num_proc (int , optional) --
  Số lượng tiến trình cho đa xử lý. Theo mặc định thì không
  sử dụng đa xử lý. batch_size trong trường hợp này mặc định là

bộ dữ liệu.config.DEFAULT_MAX_BATCH_SIZE nhưng vui lòng đặt giá trị mặc định gấp 5 hoặc 10 lần giá trị nếu bạn có đủ sức mạnh tính toán.

- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
- **to_csv_kwargs (additional keyword arguments) --
  Các tham số cần truyền tới pandas.DataFrame.to_csv của gấu trúc.
  Bây giờ, chỉ mục mặc định là Sai nếu không được chỉ định.

  If you would like to write the index, pass index=True and also set a name for the index cột theo
  vượt qua index_label .

  0 int Số ký tự hoặc byte được ghi.
  Xuất tập dữ liệu sang csv

Ví dụ:

```
>>> ds.to_csv("path/to/dataset/directory")
```

to_pandasdatasets.Dataset.to_pandashttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L5141[{"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "batched", "val": ": bool = False"}]- batch_size (int , optional) --
The size (number of rows) of the batches if batched is True .
Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .

- batched (bool) --
  Đặt thành True để trả về trình tạo tạo ra tập dữ liệu dưới dạng lô
  of batch_size rows. Defaults to False (returns the whole datasets
  once).0 pandas.DataFrame or Iterator[pandas.DataFrame]
  Trả về tập dữ liệu dưới dạng pandas.DataFrame . Cũng có thể trả lại một máy phát điện lớn bộ dữ liệu.

Ví dụ:

```
>>> ds.to_pandas()
```

to_dictdatasets.Dataset.to_dicthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/

arrow_dataset.py#L5036[{"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "batched", "val": ": bool = False"}]- batch_size (int , optional) -- The size (number of rows) of các lô nếu được phân theo đợt là True .
Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .

- batched (bool) --
  Đặt thành True để trả về trình tạo tạo ra tập dữ liệu dưới dạng lô of batch_size rows. Defaults to False (returns the whole datasets once).0 dict or Iterator[dict]
  Trả về tập dữ liệu dưới dạng lệnh Python. Cũng có thể trả về một trình tạo cho các tập dữ liệu lớn.

Ví dụ:

```
>>> ds.to_dict()
```

to_jsondatasets.Dataset.to_jsonhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L5079[{"name": "path_or_buf", "val": ": typing.Union[str, bytes, os.PathLike, typing.BinaryIO]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**to_json_kwargs", "val": ""}]- path_or_buf (PathLike or FileOrBuffer) --
Either a path to a file (e.g. file.json), a remote URI (e.g. hf://datasets/username/my_dataset_name/data.json),
hoặc BinaryIO, nơi tập dữ liệu sẽ được lưu ở định dạng đã chỉ định.

- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .
- num_proc (int , optional) --
  Số lượng tiến trình cho đa xử lý. Theo mặc định, nó không sử dụng đa xử lý. batch_size trong trường hợp này mặc định là bộ dữ liệu.config.DEFAULT_MAX_BATCH_SIZE nhưng vui lòng đặt giá trị mặc định gấp 5 hoặc 10 lần giá trị nếu bạn có đủ sức mạnh tính toán.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
- **to_json_kwargs (additional keyword arguments) --

Các tham số cần truyền tới pandas.DataFrame.to_json của gấu trúc.
Default arguments are lines=True and `orient="records".
Chỉ mục tham số mặc định là Sai nếu định hướng là "split" hoặc "table".

If you would like to write the index, pass index=True .

0 int Số ký tự hoặc byte được ghi.
Xuất tập dữ liệu sang Dòng JSON hoặc JSON.

Định dạng đầu ra mặc định là Dòng JSON.
To export to JSON, pass lines=False argument and the desired orient .

Ví dụ:

```
>>> ds.to_json("path/to/dataset/directory/filename.jsonl")
```

to_parquetdatasets.Dataset.to_parquethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L5240[{"name": "path_or_buf", "val": ": typing.Union[str, bytes, os.PathLike, typing.BinaryIO]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**parquet_writer_kwargs", "val": ""}]- path_or_buf (PathLike or FileOrBuffer) --
Either a path to a file (e.g. file.parquet), a remote URI (e.g. hf://datasets/username/my_dataset_name/data.parquet),
hoặc BinaryIO, nơi tập dữ liệu sẽ được lưu ở định dạng đã chỉ định.

- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Theo mặc định, nó nhắm đến các nhóm hàng có kích thước byte không nén tối đa là "100 MB", được xác định bởi bộ dữ liệu.config.MAX_ROW_GROUP_SIZE.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
- **parquet_writer_kwargs (additional keyword arguments) --
  Các tham số cần truyền tới pyarrow.parquet.ParquetWriter .0 int của PyArrow Số lượng ký tự hoặc byte được viết.
  Xuất tập dữ liệu sang sàn gỗ

Ví dụ:

```
>>> ds.to_parquet("path/to/dataset/directory")
```

to_sqldatasets.Dataset.to_sqlhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L5280[{"name": "name", "val": ": str"}, {"name": "con", "val": ": typing.Union[str, ForwardRef('sqlalchemy.engine.Connection'), ForwardRef('sqlalchemy.engine.Engine'), ForwardRef('sqlite3.Connection')]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**sql_writer_kwargs", "val": ""}]-
name (str) --
Tên bảng SQL.

- con (str or sqlite3.Connection or sqlalchemy.engine.Connection or
  sqlalchemy.engine.Connection) --
  Chuỗi URI hoặc đối tượng kết nối SQLite3/SQLAlchemy được sử dụng để ghi vào cơ sở dữ liệu.
- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .
- **sql_writer_kwargs (additional keyword arguments) --
  Các tham số cần truyền tới pandas.DataFrame.to_sql của gấu trúc.
  Bây giờ, chỉ mục mặc định là Sai nếu không được chỉ định.

  If you would like to write the index, pass index=True and also set a name for the index
  cột theo
  vượt qua index_label .

  0 int Số lượng bản ghi được ghi.
  Xuất tập dữ liệu sang cơ sở dữ liệu SQL.

Ví dụ:

```
>>> # con provided as a connection URI string
>>> ds.to_sql("data", "sqlite:///my_own_db.sql")
>>> # con provided as a sqlite3 connection object
>>> import sqlite3
>>> con = sqlite3.connect("my_own_db.sql")
>>> with con:
...ds.to_sql("data", con)
```

to_iterable_datasetdatasets.Dataset.to_iterable_datasethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L5372[{"name": "num_shards", "val": ": typing.Optional[int] = 1"}]- num_shards (int , default to 1) --
Số lượng phân đoạn cần xác định khi khởi tạo tập dữ liệu có thể lặp lại. Điều này đặc biệt hữu ích cho bộ dữ liệu lớn để có thể xáo trộn đúng cách,
và cũng để cho phép tải song song nhanh bằng cách sử dụng PyTorch DataLoader hoặc trong các thiết lập p
ví dụ.
Shards are defined using datasets.Dataset.shard(): it simply slices the data without writing mọi thứ trên disk.0datasets.IterableDataset
Nhận một tập dữ liệu.IterableDataset từ một tập dữ liệu kiểu bản đồ.Dataset.
This is equivalent to loading a dataset in streaming mode with datasets.load_dataset(), but nhanh hơn nhiều vì dữ liệu được truyền trực tuyến từ các tệp cục bộ.

Contrary to map-style datasets, iterable datasets are lazy and can only be iterated over (e.g. using a for loop).
Vì chúng được đọc tuần tự trong các vòng huấn luyện nên các tập dữ liệu có thể lặp lại nhanh hơn nhiều so v
tập dữ liệu phong cách.
Tất cả các phép biến đổi được áp dụng cho các tập dữ liệu có thể lặp lại như lọc hoặc xử lý đều được thực hiệ
bay khi bạn bắt đầu lặp lại tập dữ liệu.

Still, it is possible to shuffle an iterable dataset using datasets.IterableDataset.shuffle().
Đây là cách xáo trộn gần đúng nhanh, hoạt động tốt nhất nếu bạn có nhiều phân đoạn và nếu bạn chỉ định kích thước bộ đệm đủ lớn.

Để có được hiệu suất tốc độ tốt nhất, hãy đảm bảo tập dữ liệu của bạn không có ánh xạ chỉ mục.
Trong trường hợp này, dữ liệu không được đọc liền kề, đôi khi có thể bị chậm.
You can use ds = ds.flatten_indices() to write your dataset in contiguous chunks of data và có tốc độ tối ưu trước khi chuyển sang tập dữ liệu có thể lặp lại.

Ví dụ:

Cách sử dụng cơ bản:

```
>>> ids = ds.to_iterable_dataset()
>>> for example in ids:
...vượt qua
```

Với tính năng lọc và xử lý lười biếng:

```
>>> ids = ds.to_iterable_dataset()
>>> ids = ids.filter(filter_fn).map(process_fn)# will filter and process on-the-fly when you sta
>>> for example in ids:
...vượt qua
```

Với sharding để cho phép xáo trộn hiệu quả:

```
>>> ids = ds.to_iterable_dataset(num_shards=64)# the dataset is split into 64 shards to be itera
>>> ids = ids.shuffle(buffer_size=10_000)# will shuffle the shards order and use a shuffle buffe
>>> for example in ids:
...vượt qua
```

Với Trình tải dữ liệu PyTorch:

```
>>> import torch
>>> ids = ds.to_iterable_dataset(num_shards=64)
>>> ids = ids.filter(filter_fn).map(process_fn)
>>> dataloader = torch.utils.data.DataLoader(ids, num_workers=4)# will assign 64 / 4 = 16 shards
>>> for example in ids:
...vượt qua
```

Với Trình tải dữ liệu PyTorch và xáo trộn:

```
>>> import torch
>>> ids = ds.to_iterable_dataset(num_shards=64)
>>> ids = ids.shuffle(buffer_size=10_000)# will shuffle the shards order and use a shuffle buffe
>>> dataloader = torch.utils.data.DataLoader(ids, num_workers=4)# will assign 64 / 4 = 16 shards
>>> for example in ids:
...vượt qua
```

Trong thiết lập phân tán như PyTorch DDP với Trình tải dữ liệu PyTorch và xáo trộn

```
>>> from datasets.distributed import split_dataset_by_node
>>> ids = ds.to_iterable_dataset(num_shards=512)
>>> ids = ids.shuffle(buffer_size=10_000, seed=42)# will shuffle the shards order and use a shuf
>>> ids = split_dataset_by_node(ds, world_size=8, rank=0)# will keep only 512 / 8 = 64 shards fr
>>> dataloader = torch.utils.data.DataLoader(ids, num_workers=4)# will assign 64 / 4 = 16 shards
>>> for example in ids:
...vượt qua
```

Với sự xáo trộn và nhiều kỷ nguyên:

```
>>> ids = ds.to_iterable_dataset(num_shards=64)
>>> ids = ids.shuffle(buffer_size=10_000, seed=42)# will shuffle the shards order and use a shuf
>>> for epoch in range(n_epochs):
...ids.set_epoch(epoch)# will use effective_seed = seed + epoch to shuffle the shards and f
...ví dụ trong id:
...vượt qua
```

Feel free to also use `IterableDataset.set_epoch()` when using a PyTorch DataLoader or in các thiết lập phân tán.
add_faiss_indexdatasets.Dataset.add_faiss_indexhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L6110[{"name": "column", "val": ": str"}, {"name": "index_name", "val": ": typing.Optional[str] = None"}, {"name": "device", "val": ": typing.Optional[int] = None"}, {"name": "string_factory", "val": ": typing.Optional[str] = None"}, {"name": "metric_type", "val": ": typing.Optional[int] = None"}, {"name": "custom_index", "val": ": typing.Optional[ForwardRef('faiss.Index')] = None"}, {"name": "batch_size", "val": ": int = 1000"}, {"name": "train_size", "val": ": typing.Optional[int] = None"}, {"name": "faiss_verbose", "val": ": bool = False"}, {"name": "dtype", "val": " = <class 'numpy.float32'>"}]- column (str) -- Cột của các vectơ cần thêm vào chỉ mục.

- index_name (str , optional) --
  Tên_chỉ mục/mã định danh của chỉ mục.
  This is the index_name that is used to call get_nearest_examples() or search().
  Theo mặc định nó tương ứng với cột.
- device (Union[int, List[int]] , optional) --
  Nếu là số nguyên dương thì đây là chỉ số của GPU sẽ sử dụng. Nếu số nguyên âm, hãy sử dụng tất cả GP
  Nếu danh sách các số nguyên dương được chuyển vào, chỉ chạy trên các GPU đó. Theo mặc định nó sử d

CPU.

- string_factory (str , optional) --

  Điều này được chuyển đến nhà máy chỉ mục của Faiss để tạo chỉ mục.

  Lớp chỉ mục mặc định là IndexFlat .

- metric_type (int , optional) --

  Loại số liệu. Ví dụ: faiss.METRIC_INNER_PRODVEL hoặc faiss.METRIC_L2 .

- custom_index (faiss.Index , optional) --

  Chỉ mục Faiss tùy chỉnh mà bạn đã khởi tạo và định cấu hình cho nhu cầu của mình.

- batch_size (int) --

  Kích thước của lô sẽ sử dụng khi thêm vectơ vào FaissIndex . Giá trị mặc định là 1000.

- train_size (int , optional) --

  Nếu chỉ mục cần một bước huấn luyện, hãy chỉ định số lượng vectơ sẽ được sử dụng để huấn luyện chỉ số.

- faiss_verbose (bool , defaults to False) --

  Kích hoạt tính chi tiết của chỉ mục Faiss.

- dtype (data-type) --

  Dtype của mảng có nhiều mảng được lập chỉ mục.

  Mặc định là np.float32 .0

  Thêm chỉ mục dày đặc bằng Faiss để truy xuất nhanh.

  Theo mặc định, việc lập chỉ mục được thực hiện trên các vectơ của cột được chỉ định.

  You can specify device if you want to run it on GPU (device must be the GPU index).

  Bạn có thể tìm thêm thông tin về Faiss tại đây:

- Đối với nhà máy dây

Ví dụ:

```
>>> ds = datasets.load_dataset('crime_and_punish', split='train')
>>> ds_with_embeddings = ds.map(lambda example: {'embeddings': embed(example['line'])})
>>> ds_with_embeddings.add_faiss_index(column='embeddings')
>>> # query
>>> scores, retrieved_examples = ds_with_embeddings.get_nearest_examples('embeddings', embed('my n
>>> # save index
>>> ds_with_embeddings.save_faiss_index('embeddings', 'my_index.faiss')

>>> ds = datasets.load_dataset('crime_and_punish', split='train')
>>> # load index
>>> ds.load_faiss_index('embeddings', 'my_index.faiss')
>>> # query
>>> scores, retrieved_examples = ds.get_nearest_examples('embeddings', embed('my new query'), k=10
```

add_faiss_index_from_external_arraysdatasets.Dataset.add_faiss_index_from_external_array
shttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/
arrow_dataset.py#L6190[{"name": "external_arrays", "val": ": "}, {"name": "index_name", "val":
": str"}, {"name": "device", "val": ": typing.Optional[int] = None"}, {"name": "string_factory", "val":
": typing.Optional[str] = None"}, {"name": "metric_type", "val": ": typing.Optional[int] = None"},
{"name": "custom_index", "val": ": typing.Optional[ForwardRef('faiss.Index')] = None"}, {"name":
"batch_size", "val": ": int = 1000"}, {"name": "train_size", "val": ": typing.Optional[int] = None"},
{"name": "faiss_verbose", "val": ": bool = False"}, {"name": "dtype", "val": " = <class
'numpy.float32'>"}]- external_arrays (np.array) --
Nếu bạn muốn sử dụng mảng từ bên ngoài lib cho chỉ mục, bạn có thể đặt external_arrays .
Nó sẽ sử dụng external_arrays để tạo chỉ mục Faiss thay vì các mảng trong cột đã cho.

- index_name (str) --
  Tên_chỉ mục/mã định danh của chỉ mục.
  This is the index_name that is used to call get_nearest_examples() or search().
- device (Optional Union[int, List[int]] , optional) --
  Nếu là số nguyên dương thì đây là chỉ số của GPU sẽ sử dụng. Nếu số nguyên âm, hãy sử dụng tất cả GP
  Nếu danh sách các số nguyên dương được chuyển vào, chỉ chạy trên các GPU đó. Theo mặc định nó sử d
  CPU.
- string_factory (str , optional) --
  Điều này được chuyển đến nhà máy chỉ mục của Faiss để tạo chỉ mục.
  Lớp chỉ mục mặc định là IndexFlat .

- metric_type (int , optional) --
  Loại số liệu. Ví dụ: faiss.faiss.METRIC_INNER_PRODVEL hoặc faiss.METRIC_L2 .
- custom_index (faiss.Index , optional) --
  Chỉ mục Faiss tùy chỉnh mà bạn đã khởi tạo và định cấu hình cho nhu cầu của mình.
- batch_size (int , optional) --
  Kích thước của lô sẽ sử dụng khi thêm vectơ vào FaissIndex. Giá trị mặc định là 1000.

- train_size (int , optional) --
  Nếu chỉ mục cần một bước huấn luyện, hãy chỉ định số lượng vectơ sẽ được sử dụng để huấn luyện chỉ số.
- faiss_verbose (bool , defaults to False) --
  Kích hoạt tính chi tiết của chỉ mục Faiss.
- dtype (numpy.dtype) --
  Dtype của mảng có nhiều mảng được lập chỉ mục. Mặc định là np.float32.0
  Thêm chỉ mục dày đặc bằng Faiss để truy xuất nhanh.
  Chỉ mục được tạo bằng cách sử dụng các vectơ của external_arrays.
  You can specify device if you want to run it on GPU (device must be the GPU index).
  Bạn có thể tìm thêm thông tin về Faiss tại đây:
- Đối với nhà máy dây

save_faiss_indexdatasets.Dataset.save_faiss_indexhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/search.py#L535[{"name": "index_name", "val": ": str"}, {"name": "file", "val": ": typing.Union[str, pathlib.PurePath]"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}]- index_name (str) -- The index_name/identifier of the index. Đây là index_name được sử dụng để gọi .get_nearest hoặc .search .

- file (str) -- The path to the serialized faiss index on disk or remote URI (e.g. "s3://my-bucket/index.faiss").
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0
  Lưu FaissIndex vào đĩa.

Load_faiss_indexdatasets.Dataset.load_faiss_indexhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/search.py#L553[{"name": "index_name", "val": ": str"}, {"name": "file", "val": ": typing.Union[str, pathlib.PurePath]"}, {"name": "device", "val": ": typing.Union[list[int],

int, NoneType] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}]-
index_name (str) -- The index_name/identifier of the index. This is the index_name that is
đã từng
gọi .get_nearest hoặc .search .

- file (str) -- The path to the serialized faiss index on disk or remote URI (e.g.
  "s3://my-bucket/index.faiss").
- device (Optional Union[int, List[int]]) -- If positive integer, this is the index of the GPU
  để sử dụng. Nếu số nguyên âm, hãy sử dụng tất cả GPU.
  Nếu danh sách các số nguyên dương được chuyển vào, chỉ chạy trên các GPU đó. Theo mặc định nó sử d
  CPU.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0
  Tải FaissIndex từ đĩa.

Nếu bạn muốn thực hiện các cấu hình bổ sung, bạn có thể có quyền truy cập vào đối tượng chỉ mục faiss bằ
đang làm
.get_index(index_name).faiss_index to make it fit your needs.

add_elasticsearch_indexdatasets.Dataset.add_elasticsearch_indexhttps://github.com/
huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L6249[{"name": "column",
"val": ": str"}, {"name": "index_name", "val": ": typing.Optional[str] = None"}, {"name": "host",
"val": ": typing.Optional[str] = None"}, {"name": "port", "val": ": typing.Optional[int] = None"},
{"name": "es_client", "val": ": typing.Optional[ForwardRef('elasticsearch.Elasticsearch')] =
None"}, {"name": "es_index_name", "val": ": typing.Optional[str] = None"}, {"name":
"es_index_config", "val": ": typing.Optional[dict] = None"}]- column (str) --
Cột của tài liệu cần thêm vào chỉ mục.

- index_name (str , optional) --
  Tên_chỉ mục/mã định danh của chỉ mục.
  This is the index name that is used to call get_nearest_examples() or search().
  Theo mặc định nó tương ứng với cột.
- host (str , optional, defaults to localhost) --
  Máy chủ nơi ElasticSearch đang chạy.
- port (str , optional, defaults to 9200) --

Cổng nơi ElasticSearch đang chạy.

• es_client (elasticsearch.Elasticsearch , optional) --

Ứng dụng khách elaticsearch được sử dụng để tạo chỉ mục nếu máy chủ và cổng là None .

• es_index_name (str , optional) --

Tên chỉ mục elaticsearch được sử dụng để tạo chỉ mục.

• es_index_config (dict , optional) --

Cấu hình của chỉ mục elaticsearch.

Cấu hình mặc định là:

```
{
    "settings": {
        "số_of_shard": 1,
        "analysis": {"analyzer": {"stop_standard": {"type": "standard", " stopwords": "_english_"}
    },
    "mappings": {
        "properties": {
            "text": {
                "loại": "văn bản",
                "máy phân tích": "tiêu chuẩn",
                "sự tương đồng": "BM25"
            },
        }
    },
}
```</paramsdesc><paramgroups>0</paramgroups></docstring>

Thêm chỉ mục văn bản bằng cách sử dụng ElasticSearch để truy xuất nhanh. Việc này được thực hiện tại ch

<ExampleCodeBlock anchor="datasets.Dataset.add_elasticsearch_index.example">

Ví dụ:

``` con trăn
>>> es_client = elasticsearch.Elasticsearch()
>>> ds = datasets.load_dataset('crime_and_punish', split='train')
>>> ds.add_elasticsearch_index(column='line', es_client=es_client, es_index_name="my_es_index")
>>> scores, retrieved_examples = ds.get_nearest_examples('line', 'my new query', k=10)
```

Load_elasticsearch_indexdatasets.Dataset.load_elasticsearch_indexhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/search.py#L637[{"name": "index_name", "val": ": str"}, {"name": "es_index_name", "val": ": str"}, {"name": "host", "val": ": typing.Optional[str] = None"}, {"name": "port", "val": ": typing.Optional[int] = None"}, {"name": "es_client", "val": ": typing.Optional[ForwardRef('Elasticsearch')] = None"}, {"name": "es_index_config", "val": ": typing.Optional[dict] = None"}]- index_name (str) --
Tên_chỉ mục/mã định danh của chỉ mục. Đây là tên chỉ mục được sử dụng để gọi get_nearest hoặc tìm kiếm.

- es_index_name (str) --
  Tên của chỉ mục elaticsearch cần tải.
- host (str , optional, defaults to localhost) --
  Máy chủ nơi ElasticSearch đang chạy.
- port (str , optional, defaults to 9200) --
  Cổng nơi ElasticSearch đang chạy.
- es_client (elasticsearch.Elasticsearch , optional) --
  Ứng dụng khách elaticsearch được sử dụng để tạo chỉ mục nếu máy chủ và cổng là None .
- es_index_config (dict , optional) --
  Cấu hình của chỉ mục elaticsearch.
  Cấu hình mặc định là:

```
{
    "settings": {
        "số_of_shard": 1,
        "analysis": {"analyzer": {"stop_standard": {"type": "standard", " stopwords": "_english_"}
    },
    "mappings": {
        "properties": {
            "text": {
                "loại": "văn bản",
                "máy phân tích": "tiêu chuẩn",
                "sự tương đồng": "BM25"
            },
        }
    },
}
```</paramsdesc><paramgroups>0</paramgroups></docstring>
Tải chỉ mục văn bản hiện có bằng cách sử dụng ElasticSearch để truy xuất nhanh.

<docstring><name>list_indexes</name><anchor>datasets.Dataset.list_indexes</anchor><source>https://
Liệt kê `colindex_nameumns`/mã định danh của tất cả các chỉ mục đính kèm.

<docstring><name>get_index</name><anchor>datasets.Dataset.get_index</anchor><source>https://githu
Liệt kê `index_name`/mã định danh của tất cả các chỉ mục đính kèm.

<docstring><name>drop_index</name><anchor>datasets.Dataset.drop_index</anchor><source>https://g
    The `index_name`/identifier of the index.</paramsdesc><paramgroups>0</paramgroups></docstring>
Bỏ chỉ mục với cột được chỉ định.

<docstring><name>search</name><anchor>datasets.Dataset.search</anchor><source>https://github.com
    Tên/định danh của chỉ mục.
- **query** (`Union[str, np.ndarray]`) --
    Truy vấn dưới dạng một chuỗi nếu `index_name` là một chỉ mục văn bản hoặc dưới dạng một mảng có n
- **k** (`int`) --
    The number of examples to retrieve.</paramsdesc><paramgroups>0</paramgroups><rettype>`(scores,
- **scores** (`List[List[float]`): the retrieval scores from either FAISS (`IndexFlatL2` by defaul
- **indices** (`List[List[int]]`): the indices of the retrieved examples</retdesc></docstring>
Tìm các chỉ mục mẫu gần nhất trong tập dữ liệu với truy vấn.

<docstring><name>search_batch</name><anchor>datasets.Dataset.search_batch</anchor><source>https
    `index_name`/mã định danh của chỉ mục.
- **queries** (`Union[List[str], np.ndarray]`) --

Các truy vấn dưới dạng danh sách các chuỗi nếu `index_name` là một chỉ mục văn bản hoặc dưới dạng m
- **k** (`int`) --
The number of examples to retrieve per query.</paramsdesc><paramgroups>0</paramgroups><rettype
- **total_scores** (`List[List[float]`): the retrieval scores from either FAISS (`IndexFlatL2` by
- **total_indices** (`List[List[int]]`): the indices of the retrieved examples per query</retdesc>
Tìm các chỉ mục mẫu gần nhất trong tập dữ liệu với truy vấn.

<docstring><name>get_nearest_examples</name><anchor>datasets.Dataset.get_nearest_examples</anc
Index_name/mã định danh của chỉ mục.
- **query** (`Union[str, np.ndarray]`) --
Truy vấn dưới dạng một chuỗi nếu `index_name` là một chỉ mục văn bản hoặc dưới dạng một mảng có n
- **k** (`int`) --
The number of examples to retrieve.</paramsdesc><paramgroups>0</paramgroups><rettype>`(scores,
- **scores** (`List[float]`): the retrieval scores from either FAISS (`IndexFlatL2` by default) or
- **examples** (`dict`): the retrieved examples</retdesc></docstring>
Tìm các ví dụ gần nhất trong tập dữ liệu với truy vấn.

<docstring><name>get_nearest_examples_batch</name><anchor>datasets.Dataset.get_nearest_example

`index_name`/mã định danh của chỉ mục.
- **queries** (`Union[List[str], np.ndarray]`) --
Các truy vấn dưới dạng danh sách các chuỗi nếu `index_name` là một chỉ mục văn bản hoặc dưới dạng m
- **k** (`int`) --
The number of examples to retrieve per query.</paramsdesc><paramgroups>0</paramgroups><rettype
- **total_scores** (`List[List[float]`): the retrieval scores from either FAISS (`IndexFlatL2` by
- **total_examples** (`List[dict]`): the retrieved examples per query</retdesc></docstring>
Tìm các ví dụ gần nhất trong tập dữ liệu với truy vấn.

<docstring><name>info</name><anchor>datasets.Dataset.info</anchor><source>https://github.com/hug
[DatasetInfo](/docs/datasets/v4.2.0/en/package_reference/main_classes#datasets.DatasetInfo) object

<docstring><name>split</name><anchor>datasets.Dataset.split</anchor><source>https://github.com/hug
[NamedSplit](/docs/datasets/v4.2.0/en/package_reference/builder_classes#datasets.NamedSplit) objec

<docstring><name>builder_name</name><anchor>datasets.Dataset.builder_name</anchor><source>http

<docstring><name>citation</name><anchor>datasets.Dataset.citation</anchor><source>https://github.c

<docstring><name>config_name</name><anchor>datasets.Dataset.config_name</anchor><source>https

<docstring><name>dataset_size</name><anchor>datasets.Dataset.dataset_size</anchor><source>https:/

<docstring><name>description</name><anchor>datasets.Dataset.description</anchor><source>https://g

<docstring><name>download_checksums</name><anchor>datasets.Dataset.download_checksums</anch

<docstring><name>download_size</name><anchor>datasets.Dataset.download_size</anchor><source>ht

<docstring><name>features</name><anchor>datasets.Dataset.features</anchor><source>https://github.

<docstring><name>homepage</name><anchor>datasets.Dataset.homepage</anchor><source>https://gi

<docstring><name>license</name><anchor>datasets.Dataset.license</anchor><source>https://github.co

<docstring><name>size_in_bytes</name><anchor>datasets.Dataset.size_in_bytes</anchor><source>https

<docstring><name>supervised_keys</name><anchor>datasets.Dataset.supervised_keys</anchor><source

<docstring><name>version</name><anchor>datasets.Dataset.version</anchor><source>https://github.co

<docstring><name>from_csv</name><anchor>datasets.Dataset.from_csv</anchor><source>https://github
Path(s) of the CSV file(s).
- **split** ([NamedSplit](/docs/datasets/v4.2.0/en/package_reference/builder_classes#datasets.Name
Tên phân chia được gán cho tập dữ liệu.
- **features** ([Features](/docs/datasets/v4.2.0/en/package_reference/main_classes#datasets.Featur
Tính năng của bộ dữ liệu.
- **cache_dir** (`str`, *optional*, defaults to `"~/.cache/huggingface/datasets"`) --
Thư mục để lưu trữ dữ liệu.
- **keep_in_memory** (`bool`, defaults to `False`) --
Có sao chép dữ liệu trong bộ nhớ hay không.
- **num_proc** (`int`, *optional*, defaults to `None`) --
Số lượng quy trình khi tải xuống và tạo tập dữ liệu cục bộ.
Điều này hữu ích nếu tập dữ liệu được tạo thành từ nhiều tệp. Đa xử lý bị tắt theo mặc định

<Added version="2.8.0"/>
- ****kwargs** (additional keyword arguments) --
Keyword arguments to be passed to `pandas.read_csv`.</paramsdesc><paramgroups>0</paramgroups>
Create Dataset from CSV file(s).

<ExampleCodeBlock anchor="datasets.Dataset.from_csv.example">

Ví dụ:

``` py
>>> ds = Dataset.from_csv('path/to/dataset.csv')
```

from_jsondatasets.Dataset.from_jsonhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1189[{"name": "path_or_paths", "val": ": typing.Union[str, bytes, os.PathLike, list[typing.Union[str, bytes, os.PathLike]]]"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "features", "val": ":

typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "field", "val": ": typing.Optional[str] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "**kwargs", "val": ""}]- path_or_paths (path-like or list of path-like) --
Path(s) of the JSON or JSON Lines file(s).

- split (NamedSplit, optional) --
  Tên phân chia được gán cho tập dữ liệu.
- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str , optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- field (str , optional) --
  Tên trường của tệp JSON chứa tập dữ liệu.
- num_proc (int , optional defaults to None) --
  Số lượng quy trình khi tải xuống và tạo tập dữ liệu cục bộ.
  Điều này hữu ích nếu tập dữ liệu được tạo thành từ nhiều tệp. Đa xử lý bị vô hiệu hóa bởi mặc định.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển tới JsonConfig .0Dataset
  Create Dataset from JSON or JSON Lines file(s).

Ví dụ:

```
>>> ds = Dataset.from_json('path/to/dataset.json')
```

from_parquetdatasets.Dataset.from_parquethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1246[{"name": "path_or_paths", "val": ": typing.Union[str, bytes, os.PathLike, list[typing.Union[str, bytes, os.PathLike]]]"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "columns", "val": ": typing.Optional[list[str]] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "**kwargs", "val": ""}]- path_or_paths (path-like or list of path-like) --

Path(s) of the Parquet file(s).

- split (NamedSplit , optional) --
  Tên phân chia được gán cho tập dữ liệu.
- features (Features , optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str , optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- columns (List[str] , optional) --
  Nếu không phải None , chỉ những cột này sẽ được đọc từ tệp.
  Tên cột có thể là tiền tố của trường lồng nhau, ví dụ: 'a' sẽ chọn
  'a.b', 'a.c' và 'a.d.e'.
- num_proc (int , optional, defaults to None) --
  Số lượng quy trình khi tải xuống và tạo tập dữ liệu cục bộ.
  Điều này hữu ích nếu tập dữ liệu được tạo thành từ nhiều tệp. Đa xử lý bị vô hiệu hóa bởi mặc định.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển đến ParquetConfig .0Dataset
  Create Dataset from Parquet file(s).

Ví dụ:

```
>>> ds = Dataset.from_parquet('path/to/dataset.parquet')
```

from_textdatasets.Dataset.from_texthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1305[{"name": "path_or_paths", "val": ": typing.Union[str, bytes, os.PathLike, list[typing.Union[str, bytes, os.PathLike]]]"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "**kwargs", "val": ""}]- path_or_paths (path-like or list of path-like) --
Path(s) of the text file(s).

- split (NamedSplit , optional) --
  Tên phân chia được gán cho tập dữ liệu.
- features (Features , optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str , optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- num_proc (int , optional, defaults to None) --
  Số lượng quy trình khi tải xuống và tạo tập dữ liệu cục bộ.
  Điều này hữu ích nếu tập dữ liệu được tạo thành từ nhiều tệp. Đa xử lý bị vô hiệu hóa bởi mặc định.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển tới TextConfig .0Dataset
  Create Dataset from text file(s).

Ví dụ:

```
>>> ds = Dataset.from_text('path/to/dataset.txt')
```

from_sqldatasets.Dataset.from_sqlhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L1420[{"name": "sql", "val": ": typing.Union[str, ForwardRef('sqlalchemy.sql.Selectable')]"}, {"name": "con", "val": ": typing.Union[str, ForwardRef('sqlalchemy.engine.Connection'), ForwardRef('sqlalchemy.engine.Engine'), ForwardRef('sqlite3.Connection')]"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "**kwargs", "val": ""}]- sql (str or sqlalchemy.sql.Selectable) --
Truy vấn SQL sẽ được thực thi hoặc tên bảng.

- con (str or sqlite3.Connection or sqlalchemy.engine.Connection or sqlalchemy.engine.Connection) --
  Chuỗi URI được sử dụng để khởi tạo kết nối cơ sở dữ liệu hoặc SQLite3/SQLAlchemy đối tượng kết nối.
- features (Features, optional) --
  Tính năng của bộ dữ liệu.

- cache_dir (str , optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển tới SqlConfig .0Dataset
  Tạo Tập dữ liệu từ truy vấn SQL hoặc bảng cơ sở dữ liệu.

Ví dụ:

```
>>> # Fetch a database table
>>> ds = Dataset.from_sql("test_data", "postgres:///db_name")
>>> # Execute a SQL query on the table
>>> ds = Dataset.from_sql("SELECT sentence FROM test_data", "postgres:///db_name")
>>> # Use a Selectable object to specify the query
>>> from sqlalchemy import select, text
>>> stmt = select([text("sentence")]).select_from(text("test_data"))
>>> ds = Dataset.from_sql(stmt, "postgres:///db_name")
```

[!TIP]
Tập dữ liệu được trả về chỉ có thể được lưu vào bộ đệm nếu con được chỉ định làm chuỗi URI.

Align_labels_with_mappingdatasets.Dataset.align_labels_with_mappinghttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L6371[{"name": "label2id", "val": ": dict"}, {"name": "label_column", "val": ": str"}]- label2id (dict) --
Tên nhãn để ánh xạ ID để căn chỉnh tập dữ liệu.

- label_column (str) --
  Tên cột của nhãn cần căn chỉnh trên.0
  Căn chỉnh ánh xạ ID nhãn và tên nhãn của tập dữ liệu để khớp với ánh xạ label2id đầu vào.
  Điều này hữu ích khi bạn muốn đảm bảo rằng các nhãn dự đoán của mô hình được căn chỉnh theo tập dữ liệu.
  Việc căn chỉnh được thực hiện bằng cách sử dụng tên nhãn chữ thường.

Ví dụ:

```
>>> # dataset with mapping {'entailment': 0, 'neutral': 1, 'contradiction': 2}
>>> ds = load_dataset("nyu-mll/glue", "mnli", split="train")
>>> # mapping to align with
>>> label2id = {'CONTRADICTION': 0, 'NEUTRAL': 1, 'ENTAILMENT': 2}
>>> ds_aligned = ds.align_labels_with_mapping(label2id, "label")
```

bộ dữ liệu.concatenate_datasetsdatasets.concatenate_datasetshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/combine.py#L166[{"name": "dsets", "val": ": list"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "axis", "val": ": int = 0"}]- dsets (List[datasets.Dataset]) --
Danh sách các bộ dữ liệu cần nối.

- info (DatasetInfo , optional) --
  Thông tin về tập dữ liệu, như mô tả, trích dẫn, v.v.
- split (NamedSplit , optional) --
  Tên của tập dữ liệu được chia.
- axis ({0, 1} , defaults to 0) --
  Axis to concatenate over, where 0 means over rows (vertically) and 1 means over
  cột
  (horizontally).
  0

Chuyển đổi danh sách Tập dữ liệu có cùng lược đồ thành một Tập dữ liệu duy nhất.

Ví dụ:

```
>>> ds3 = concatenate_datasets([ds1, ds2])
```

bộ dữ liệu.interleave_datasetsdatasets.interleave_datasetshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/combine.py#L18[{"name": "datasets", "val": ": list"}, {"name": "probabilities", "val": ": typing.Optional[list[float]] = None"}, {"name": "seed", "val": ": typing.Optional[int] = None"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "stopping_strategy", "val": ": typing.Literal['first_exhausted', 'all_exhausted', 'all_exhausted_without_replacement'] = 'first_exhausted'"}]- datasets (List[Dataset] or

List[IterableDataset]) --
Danh sách các tập dữ liệu để xen kẽ.

- probabilities (List[float] , optional, defaults to None) --
  Nếu được chỉ định, tập dữ liệu mới được xây dựng bằng cách lấy mẫu
  ví dụ từ một nguồn tại một thời điểm theo các xác suất này.
- seed (int , optional, defaults to None) --
  Hạt giống ngẫu nhiên được sử dụng để chọn nguồn cho mỗi ví dụ.
- info (DatasetInfo, optional) --
  Thông tin về tập dữ liệu, như mô tả, trích dẫn, v.v.

- split (NamedSplit, optional) --
  Tên của tập dữ liệu được chia.

- stopping_strategy (str , defaults to first_exhausted) --
  Ba chiến lược được đề xuất ngay bây giờ, first_exhausted, all_exhausted và
  all_exhausted_without_replacement .
  Theo mặc định, first_exhaused là một chiến lược lấy mẫu dưới mức, tức là việc xây dựng tập dữ liệu là
  dừng ngay khi một tập dữ liệu hết mẫu.
  Nếu chiến lược là all_exhausted , chúng tôi sẽ sử dụng chiến lược lấy mẫu quá mức, tức là tập dữ liệu
  quá trình xây dựng bị dừng ngay khi mọi mẫu của mọi tập dữ liệu được thêm vào
  ít nhất một lần.
  Khi chiến lược là all_exhausted_without_replacement, chúng tôi đảm bảo rằng mỗi mẫu trong
  mỗi tập dữ liệu chỉ được lấy mẫu một lần.
  Lưu ý rằng nếu chiến lược là all_exhausted thì kích thước tập dữ liệu xen kẽ có thể rất lớn:
    không có xác suất, tập dữ liệu kết quả sẽ có max_length_datasets*nb_dataset
      mẫu.
    với xác suất cho trước, tập dữ liệu thu được sẽ có nhiều mẫu hơn nếu một số tập dữ liệu
      có xác suất truy cập thực sự thấp.0Dataset hoặc IterableDatasetReturn loại tùy thuộc vào
      trên tập dữ liệu đầu vào
      tham số. Tập dữ liệu nếu đầu vào là danh sách Tập dữ liệu, IterableDataset nếu đầu vào là một
      danh sách
      IterableDataset .

Interleave several datasets (sources) into a single dataset.
Tập dữ liệu mới được xây dựng bằng cách xen kẽ giữa các nguồn để lấy ví dụ.

Bạn có thể sử dụng hàm này trên danh sách các đối tượng Dataset hoặc trên danh sách các đối tượng Iterab

- If probabilities is None (default) the new dataset is constructed by cycling between mỗi nguồn để lấy ví dụ.
- Nếu xác suất không phải là None , tập dữ liệu mới được xây dựng bằng cách lấy các ví dụ từ một nguồn ngẫu nhiên tại một thời điểm theo xác suất được cung cấp.

Tập dữ liệu kết quả kết thúc khi một trong các tập dữ liệu nguồn hết mẫu trừ khi lấy mẫu quá mức là Đúng,
trong trường hợp đó, tập dữ liệu kết quả kết thúc khi ít nhất tất cả các tập dữ liệu đã hết ví dụ một lần.

Lưu ý đối với bộ dữ liệu có thể lặp lại:

Trong thiết lập phân tán hoặc trong PyTorch DataLoader, chiến lược dừng được áp dụng cho mỗi quá trình.
Do đó, chiến lược "first_exhausted" trên tập dữ liệu có thể lặp lại được phân đoạn có thể tạo ra ít hơn samples in total (up to 1 missing sample per subdataset per worker).

Ví dụ:

For regular datasets (map-style):

```
>>> from datasets import Dataset, interleave_datasets
>>> d1 = Dataset.from_dict({"a": [0, 1, 2]})
>>> d2 = Dataset.from_dict({"a": [10, 11, 12]})
>>> d3 = Dataset.from_dict({"a": [20, 21, 22]})
>>> dataset = interleave_datasets([d1, d2, d3], probabilities=[0.7, 0.2, 0.1], seed=42, stopping_s
>>> dataset["a"]
[10, 0, 11, 1, 2, 20, 12, 10, 0, 1, 2, 21, 0, 11, 1, 2, 0, 1, 12, 2, 10, 0, 22]
>>> dataset = interleave_datasets([d1, d2, d3], probabilities=[0.7, 0.2, 0.1], seed=42)
>>> dataset["a"]
[10, 0, 11, 1, 2]
>>> dataset = interleave_datasets([d1, d2, d3])
>>> dataset["a"]
[0, 10, 20, 1, 11, 21, 2, 12, 22]
>>> dataset = interleave_datasets([d1, d2, d3], stopping_strategy="all_exhausted")
>>> dataset["a"]
[0, 10, 20, 1, 11, 21, 2, 12, 22]
>>> d1 = Dataset.from_dict({"a": [0, 1, 2]})
>>> d2 = Dataset.from_dict({"a": [10, 11, 12, 13]})
>>> d3 = Dataset.from_dict({"a": [20, 21, 22, 23, 24]})
>>> dataset = interleave_datasets([d1, d2, d3])
>>> dataset["a"]
[0, 10, 20, 1, 11, 21, 2, 12, 22]
>>> dataset = interleave_datasets([d1, d2, d3], stopping_strategy="all_exhausted")
>>> dataset["a"]
[0, 10, 20, 1, 11, 21, 2, 12, 22, 0, 13, 23, 1, 10, 24]
>>> dataset = interleave_datasets([d1, d2, d3], probabilities=[0.7, 0.2, 0.1], seed=42)
>>> dataset["a"]
[10, 0, 11, 1, 2]
>>> dataset = interleave_datasets([d1, d2, d3], probabilities=[0.7, 0.2, 0.1], seed=42, stopping_s
>>> dataset["a"]
[10, 0, 11, 1, 2, 20, 12, 13, ..., 0, 1, 2, 0, 24]
For datasets in streaming mode (iterable):

>>> from datasets import interleave_datasets
>>> d1 = load_dataset('allenai/c4', 'es', split='train', streaming=True)
>>> d2 = load_dataset('allenai/c4', 'fr', split='train', streaming=True)
>>> dataset = interleave_datasets([d1, d2])
>>> iterator = iter(dataset)
>>> next(iterator)
```

{'text': 'Comprar Zapatillas para niña en chancla con goma por...'}
>>> next(iterator)
{'text': 'Le sacre de philippe ier, 23 mai 1059 - Compte Rendu...'

bộ dữ liệu.distributed.split_dataset_by_nodedatasets.distributed.split_dataset_by_nodehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/distributed.py#L10[{"name": "dataset", "val": ": ~DatasetType"}, {"name": "rank", "val": ": int"}, {"name": "world_size", "val": ": int"}]-
dataset (Dataset or IterableDataset) --
Tập dữ liệu được phân chia theo nút.

- rank (int) --
  Thứ hạng của nút hiện tại.
- world_size (int) --
  Tổng số nút.0Dataset hoặc IterableDatasetBộ dữ liệu sẽ được sử dụng trên nút tại xếp hạng.

Tách tập dữ liệu cho nút ở thứ hạng xếp hạng trong nhóm nút có kích thước world_size .

Đối với bộ dữ liệu kiểu bản đồ:

Mỗi nút được gán một đoạn dữ liệu, ví dụ: hạng 0 được cấp đoạn đầu tiên của tập dữ liệu.
Để tối đa hóa thông lượng tải dữ liệu, các khối được tạo từ dữ liệu liền kề trên đĩa nếu có thể.

Đối với bộ dữ liệu có thể lặp lại:

If the dataset has a number of shards that is a factor of world_size (i.e. if
dataset.num_shards % world_size == 0),
sau đó các phân đoạn được phân bổ đều trên các nút, được tối ưu hóa nhất.
Mặt khác, mỗi nút giữ 1 ví dụ ngoài world_size, bỏ qua các ví dụ khác.

bộ dữ liệu.enable_cachingdatasets.enable_cachinghttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/fingerprint.py#L94[]

Khi áp dụng các phép biến đổi trên tập dữ liệu, dữ liệu sẽ được lưu trữ trong các tệp bộ đệm.
Cơ chế bộ đệm cho phép tải lại tệp bộ đệm hiện có nếu nó đã được tính toán.

Có thể tải lại tập dữ liệu vì các tệp bộ đệm được đặt tên bằng dấu vân tay của tập dữ liệu,
được cập nhật
sau mỗi lần biến đổi.

Nếu bị tắt, thư viện sẽ không còn tải lại các tệp dữ liệu được lưu trong bộ nhớ đệm khi áp dụng các phép biế
các tập dữ liệu.
Chính xác hơn, nếu bộ nhớ đệm bị tắt:

- các tập tin bộ nhớ đệm luôn được tạo lại
- các tập tin bộ đệm được ghi vào một thư mục tạm thời sẽ bị xóa khi phiên đóng
- các tập tin bộ nhớ đệm được đặt tên bằng cách sử dụng hàm băm ngẫu nhiên thay vì dấu vân tay của tập
- use save_to_disk() to save a transformed dataset or it will be deleted when session closes
- caching doesn't affect load_dataset(). If you want to regenerate a dataset from scratch you
  nên sử dụng
  the download_mode parameter in load_dataset().

bộ dữ liệu.disable_cachingdatasets.disable_cachinghttps://github.com/huggingface/datasets/
blob/4.2.0/src/datasets/fingerprint.py#L115[]

Khi áp dụng các phép biến đổi trên tập dữ liệu, dữ liệu sẽ được lưu trữ trong các tệp bộ đệm.
Cơ chế bộ đệm cho phép tải lại tệp bộ đệm hiện có nếu nó đã được tính toán.

Có thể tải lại tập dữ liệu vì các tệp bộ đệm được đặt tên bằng dấu vân tay của tập dữ liệu,
được cập nhật
sau mỗi lần biến đổi.

Nếu bị tắt, thư viện sẽ không còn tải lại các tệp dữ liệu được lưu trong bộ nhớ đệm khi áp dụng các phép biế
các tập dữ liệu.
Chính xác hơn, nếu bộ nhớ đệm bị tắt:

- các tập tin bộ nhớ đệm luôn được tạo lại
- các tập tin bộ đệm được ghi vào một thư mục tạm thời sẽ bị xóa khi phiên đóng
- các tập tin bộ nhớ đệm được đặt tên bằng cách sử dụng hàm băm ngẫu nhiên thay vì dấu vân tay của tập
- use save_to_disk() to save a transformed dataset or it will be deleted when session closes
- caching doesn't affect load_dataset(). If you want to regenerate a dataset from scratch you
  nên sử dụng
  the download_mode parameter in load_dataset().

bộ dữ liệu.is_caching_enableddatasets.is_caching_enabledhttps://github.com/huggingface/
datasets/blob/4.2.0/src/datasets/fingerprint.py#L136[]

Khi áp dụng các phép biến đổi trên tập dữ liệu, dữ liệu sẽ được lưu trữ trong các tệp bộ đệm.

Cơ chế bộ đệm cho phép tải lại tệp bộ đệm hiện có nếu nó đã được tính toán.

Có thể tải lại tập dữ liệu vì các tệp bộ đệm được đặt tên bằng dấu vân tay của tập dữ liệu,
được cập nhật
sau mỗi lần biến đổi.

Nếu bị tắt, thư viện sẽ không còn tải lại các tệp dữ liệu được lưu trong bộ nhớ đệm khi áp dụng các phép biế
các tập dữ liệu.
Chính xác hơn, nếu bộ nhớ đệm bị tắt:

- các tập tin bộ nhớ đệm luôn được tạo lại
- các tập tin bộ đệm được ghi vào một thư mục tạm thời sẽ bị xóa khi phiên đóng
- các tập tin bộ nhớ đệm được đặt tên bằng cách sử dụng hàm băm ngẫu nhiên thay vì dấu vân tay của tập
- use save_to_disk()] to save a transformed dataset or it will be deleted when session
  đóng cửa
- caching doesn't affect load_dataset(). If you want to regenerate a dataset from scratch you
  nên sử dụng
  the download_mode parameter in load_dataset().

tập dữ liệu lớp.Columndatasets.Columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/
datasets/arrow_dataset.py#L633[{"name": "source", "val": ":
typing.Union[ForwardRef('Dataset'), ForwardRef('Column')]"}, {"name": "column_name", "val":
": str"}]

Một lần lặp cho một cột cụ thể của Tập dữ liệu.

Ví dụ:

Lặp lại các văn bản của cột "văn bản" của tập dữ liệu:

```
for text in dataset["text"]:
    None
```

Nó cũng hoạt động với các cột lồng nhau:

```
for source in dataset["metadata"]["source"]:
    None
```

DatasetDictdatasets.DatasetDict

Dictionary with split names as keys ('train', 'test' for example), and Dataset objects as values.
Nó cũng có các phương thức chuyển đổi tập dữ liệu như bản đồ hoặc bộ lọc, để xử lý tất cả các phần tách cù

bộ dữ liệu lớp.DatasetDictdatasets.DatasetDicthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L57""
A dictionary (dict of str: datasets.Dataset) with dataset transforms methods (map, filter, etc.)

datadatasets.DatasetDict.datahttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L98[]
Các bảng Mũi tên Apache hỗ trợ mỗi phần tách.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.data
```

cache_filesdatasets.DatasetDict.cache_fileshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L113[]
Các tệp bộ đệm chứa bảng Mũi tên Apache sao lưu mỗi phần tách.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.cache_files
{'test': [{'filename': '/root/.cache/huggingface/datasets/rotten_tomatoes_movie_review/default/1.0
  'train': [{'filename': '/root/.cache/huggingface/datasets/rotten_tomatoes_movie_review/default/1.
  'validation': [{'filename': '/root/.cache/huggingface/datasets/rotten_tomatoes_movie_review/defau
```

num_columnsdatasets.DatasetDict.num_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L131[]
Số cột trong mỗi phần của tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.num_columns
{'test': 2, 'train': 2, 'validation': 2}
```

num_rowsdatasets.DatasetDict.num_rowshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L147[]
Số lượng hàng trong mỗi phần của tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.num_rows
{'test': 1066, 'train': 8530, 'validation': 1066}
```

cột_namesdatasets.DatasetDict.column_nameshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L163[]
Tên của các cột trong mỗi phần của tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.column_names
{'test': ['text', 'label'],
 'train': ['text', 'label'],
 'validation': ['text', 'label']}
```

Shapedatasets.DatasetDict.shapehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L181[]
Shape of each split of the dataset (number of rows, number of columns).

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.shape
{'test': (1066, 2), 'train': (8530, 2), 'validation': (1066, 2)}
```

Uniquedatasets.DatasetDict.uniquehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L230[{"name": "column", "val": ": str"}]- column (str) -- column name (list all the column names with column_names)0Dict[str , list]Dictionary of các phần tử duy nhất trong cột nhất định.
Trả về danh sách các phần tử duy nhất trong một cột cho mỗi phần tách.

Điều này được thực hiện ở phần phụ trợ cấp thấp và như vậy rất nhanh.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.unique("label")
{'test': [1, 0], 'train': [1, 0], 'validation': [1, 0]}
```

cleanup_cache_filesdatasets.DatasetDict.cleanup_cache_fileshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L254[] Dict with the number of removed files cho mỗi lần chia
Dọn dẹp tất cả các tệp bộ đệm trong thư mục bộ đệm của tập dữ liệu, ngoại trừ tệp bộ đệm hiện đang được có một cái.
Hãy cẩn thận khi chạy lệnh này vì không có tiến trình nào khác hiện đang sử dụng bộ đệm khác tập tin.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.cleanup_cache_files()
{'test': 0, 'train': 0, 'validation': 0}
```

mapdatasets.DatasetDict.maphttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L818[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"},

{"name": "with_indices", "val": ": bool = False"}, {"name": "with_rank", "val": ": bool = False"}, {"name": "with_split", "val": ": bool = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "drop_last_batch", "val": ": bool = False"}, {"name": "remove_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "cache_file_names", "val": ": typing.Optional[dict[str, typing.Optional[str]]] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "disable_nullable", "val": ": bool = False"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "desc", "val": ": typing.Optional[str] = None"}, {"name": "try_original_type", "val": ": typing.Optional[bool] = True"}]- function (callable) -- with one of the following signature:

- function(example: Dict[str, Any]) -> Dict[str, Any] if batched=False and with_indices=False
- function(example: Dict[str, Any], indices: int) -> Dict[str, Any] if batched=False and with_indices=True
- function(batch: Dict[str, list]) -> Dict[str, list] if batched=True and with_indices=False
- function(batch: Dict[str, list], indices: list[int]) -> Dict[str, list] if batched=True and with_indices=True

Để sử dụng nâng cao, hàm cũng có thể trả về pyarrow.Table .
Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song.
Moreover if your function returns nothing (None), then map will run your function and return tập dữ liệu không thay đổi
Nếu không có chức năng nào được cung cấp, mặc định là chức năng nhận dạng: lambda x: x .

- with_indices (bool , defaults to False) -- Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này chữ ký của hàm should be def function(example, idx): ... .
- with_rank (bool , defaults to False) -- Cung cấp thứ hạng quy trình để hoạt động. Lưu ý rằng trong trường hợp này signature of function should be def function(example[, idx], rank): ... .

• with_split (bool , defaults to False) --
  Cung cấp quá trình phân chia thành chức năng. Lưu ý rằng trong trường hợp này
  signature of function should be def function(example[, idx], split): ... .
• input_columns ([Union[str, list[str]]] , optional, defaults to None) --
  Các cột được chuyển vào chức năng như
  những lập luận mang tính vị trí. Nếu Không, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được ch
  lý lẽ.
• batched (bool , defaults to False) --
  Cung cấp hàng loạt ví dụ để hoạt động.
• batch_size (int , optional, defaults to 1000) --
  Number of examples per batch provided to function if batched=True ,
  batch_size <= 0 or batch_size == None then provide the full dataset as a single batch to
  chức năng .
• drop_last_batch (bool , defaults to False) --
  Liệu lô cuối cùng có nhỏ hơn batch_size hay không
  bị loại bỏ thay vì được xử lý bởi hàm.
• remove_columns ([Union[str, list[str]]] , optional, defaults to None) --
  Xóa vùng chọn cột trong khi thực hiện ánh xạ.
  Các cột sẽ bị xóa trước khi cập nhật các ví dụ với đầu ra của hàm , tức là nếu
  chức năng đang thêm
  các cột có tên trong Remove_columns , các cột này sẽ được giữ lại.
• keep_in_memory (bool , defaults to False) --
  Giữ tập dữ liệu trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
• load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ tính toán hiện tại từ hàm
  có thể được xác định, hãy sử dụng nó thay vì tính toán lại.
• cache_file_names ([Dict[str, str]] , optional, defaults to None) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  kết quả tính toán thay vì tên tệp bộ đệm được tạo tự động.
  Bạn phải cung cấp một cache_file_name cho mỗi tập dữ liệu trong từ điển tập dữ liệu.
• writer_batch_size (int , default 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
  tốc độ xử lý.
  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn

bộ nhớ tạm thời trong khi chạy bản đồ.

- features ([datasets.Features] , optional, defaults to None) --
  Sử dụng một Tính năng cụ thể để lưu trữ tệp bộ đệm
  thay vì cái được tạo tự động.
- disable_nullable (bool , defaults to False) --
  Không cho phép giá trị null trong bảng.
- fn_kwargs (Dict , optional, defaults to None) --
  Đối số từ khóa được truyền cho hàm
- num_proc (int , optional, defaults to None) --
  Số lượng tiến trình được sử dụng cho đa xử lý.
    Nếu Không có hoặc 0 , thì không có đa xử lý nào được sử dụng và thao tác sẽ chạy trong quy trình chí
    Nếu lớn hơn 1 , một hoặc nhiều quy trình công nhân được sử dụng để xử lý dữ liệu trong
      song song.
      Note: The function passed to map() must be picklable for multiprocessing to work
      chính xác
      (i.e., prefer functions defined at the top level of a module, not inside another function
      or class).
- desc (str , optional, defaults to None) --
  Mô tả có ý nghĩa sẽ được hiển thị cùng với thanh tiến trình trong khi lập bản đồ
  ví dụ.
- try_original_type (Optional[bool] , defaults to True) --
  Try to keep the types of the original columns (e.g. int32 -> int32).
  Đặt thành Sai nếu bạn muốn luôn suy ra các loại mới.0

Apply a function to all the examples in the table (individually or in batches) and update the
bàn.
Nếu hàm của bạn trả về một cột đã tồn tại thì nó sẽ ghi đè lên cột đó.
Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Bạn có thể chỉ định xem hàm có nên được bó hay không bằng tham số được bó:

- Nếu bó là Sai thì hàm sẽ lấy 1 mẫu và trả về 1 mẫu.
  An example is a dictionary, e.g. {"text": "Hello there !"} .
- Nếu batched là True và batch_size là 1 thì hàm sẽ lấy một batch gồm 1 ví dụ như
  đầu vào và có thể trả về một lô có 1 hoặc nhiều mẫu.
  A batch is a dictionary, e.g. a batch of 1 example is {"text": ["Hello there !"]} .

- If batched is True and batch_size is n > 1 , then the function takes a batch of n
  ví dụ làm đầu vào và có thể trả về một lô có n ví dụ hoặc với số lượng tùy ý
  ví dụ.
  Lưu ý rằng đợt cuối cùng có thể có ít hơn n mẫu.
  A batch is a dictionary, e.g. a batch of n examples is {"text": ["Hello there !"] * n} .

Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song, với tối đa một nghìn cuộc gọi đồng thời.
Bạn nên sử dụng asyncio.Semaphore trong hàm của mình nếu bạn muốn đặt mức tối đa số thao tác có thể thực hiện đồng thời.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> def add_prefix(example):
...example["text"] = "Review: " + example["text"]
...trả lại ví dụ
>>> ds = ds.map(add_prefix)
>>> ds["train"][0:3]["text"]
['Review: the rock is destined to be the 21st century's new " conan " and that he's going to make
 'Đánh giá: phần tiếp theo được xây dựng công phu tuyệt đẹp của bộ ba phim "chúa tể của những chiếc nh
 'Review: effective but too-tepid biopic']

# xử lý một loạt ví dụ
>>> ds = ds.map(lambda example: tokenizer(example["text"]), batched=True)
# đặt số lượng bộ xử lý
>>> ds = ds.map(add_prefix, num_proc=4)
```

filterdatasets.DatasetDict.filterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L979[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "with_indices", "val": ": bool = False"}, {"name": "with_rank", "val": ": bool = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "cache_file_names", "val": ": typing.Optional[dict[str, typing.Optional[str]]] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] =

1000"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "desc", "val": ": typing.Optional[str] = None"}]- function (Callable) -- Callable with one of the following signatures:

- function(example: Dict[str, Any]) -> bool if batched=False and with_indices=False and with_rank=False
- function(example: Dict[str, Any], *extra_args) -> bool if batched=False and with_indices=True and/or with_rank=True (one extra arg for each)
- function(batch: Dict[str, list]) -> list[bool] if batched=True and with_indices=False and with_rank=False
- function(batch: Dict[str, list], *extra_args) -> list[bool] if batched=True and with_indices=True and/or with_rank=True (one extra arg for each)

Nếu không có hàm nào được cung cấp, mặc định là hàm luôn True: lambda x: True .

- with_indices (bool , defaults to False) --
  Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này signature of function should be def function(example, idx[, rank]): ... .
- with_rank (bool , defaults to False) --
  Cung cấp thứ hạng quy trình để hoạt động. Lưu ý rằng trong trường hợp này signature of function should be def function(example[, idx], rank): ... .
- input_columns ([Union[str, list[str]]] , optional, defaults to None) --
  Các cột được chuyển vào chức năng như những lập luận mang tính vị trí. Nếu Không, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được ch lý lẽ.
- batched (bool , defaults to False) --
  Cung cấp hàng loạt ví dụ để hoạt động.
- batch_size (int , optional, defaults to 1000) --
  Number of examples per batch provided to function if batched=True batch_size <= 0 or batch_size == None then provide the full dataset as a single batch to chức năng .
- keep_in_memory (bool , defaults to False) --
  Giữ tập dữ liệu trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ tính toán hiện tại từ hàm có thể được xác định, hãy sử dụng nó thay vì tính toán lại.

• cache_file_names ([Dict[str, str]] , optional, defaults to None) --
Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
kết quả tính toán thay vì tên tệp bộ đệm được tạo tự động.
Bạn phải cung cấp một cache_file_name cho mỗi tập dữ liệu trong từ điển tập dữ liệu.
• writer_batch_size (int , defaults to 1000) --
Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và
tốc độ xử lý.
Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn
bộ nhớ tạm thời trong khi chạy bản đồ.
• fn_kwargs (Dict , optional, defaults to None) --
Đối số từ khóa được truyền cho hàm
• num_proc (int , optional, defaults to None) --
Số lượng tiến trình được sử dụng cho đa xử lý.
Nếu Không có hoặc 0 , thì không có đa xử lý nào được sử dụng và thao tác sẽ chạy trong quy trình chí
Nếu lớn hơn 1 , một hoặc nhiều quy trình công nhân được sử dụng để xử lý dữ liệu trong
song song.
Note: The function passed to map() must be picklable for multiprocessing to work
chính xác
(i.e., prefer functions defined at the top level of a module, not inside another function
or class).
• desc (str , optional, defaults to None) --
Mô tả có ý nghĩa sẽ được hiển thị cùng với thanh tiến trình trong khi lọc
ví dụ.0
Áp dụng chức năng lọc cho tất cả các phần tử trong bảng theo đợt
và cập nhật bảng để tập dữ liệu chỉ bao gồm các ví dụ theo bộ lọc
chức năng.
Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.filter(lambda x: x["label"] == 1)
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 4265
    })
    validation: Dataset({
        features: ['text', 'label'],
        số_hàng: 533
    })
    test: Dataset({
        features: ['text', 'label'],
        số_hàng: 533
    })
})
```

Sortdatasets.DatasetDict.sorthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1144[{"name": "column_names", "val": ": typing.Union[str, collections.abc.Sequence[str]]"}, {"name": "reverse", "val": ": typing.Union[bool, collections.abc.Sequence[bool]] = False"}, {"name": "null_placement", "val": ": str = 'at_end'"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "indices_cache_file_names", "val": ": typing.Optional[dict[str, typing.Optional[str]]] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}]- column_names (Union[str, Sequence[str]]) -- Column name(s) to sort by.

- reverse (Union[bool, Sequence[bool]] , defaults to False) --
  Nếu Đúng, hãy sắp xếp theo thứ tự giảm dần thay vì tăng dần. Nếu một bool duy nhất được cung cấp, giá trị được áp dụng để sắp xếp tất cả các tên cột. Nếu không thì danh sách các bools có
  Phải cung cấp cùng độ dài và thứ tự như tên cột.
- null_placement (str , defaults to at_end) --
  Đặt giá trị Không có ở đầu nếu at_start hoặc đầu tiên hoặc ở cuối nếu at_end hoặc cuối cùng
- keep_in_memory (bool , defaults to False) --
  Giữ các chỉ mục đã sắp xếp trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --

Nếu một tệp bộ đệm lưu trữ các chỉ mục được sắp xếp
có thể được xác định, hãy sử dụng nó thay vì tính toán lại.

- indices_cache_file_names ([Dict[str, str]] , optional, defaults to None) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các
  ánh xạ chỉ mục thay vì tên tệp bộ đệm được tạo tự động.
  Bạn phải cung cấp một cache_file_name cho mỗi tập dữ liệu trong từ điển tập dữ liệu.
- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị cao hơn mang lại tệp bộ nhớ đệm nhỏ hơn, giá trị thấp hơn tiêu tốn ít bộ nhớ tạm thời hơn.0
  Tạo tập dữ liệu mới được sắp xếp theo một hoặc nhiều cột.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset('cornell-movie-review-data/rotten_tomatoes')
>>> ds['train']['label'][:10]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
>>> sorted_ds = ds.sort('label')
>>> sorted_ds['train']['label'][:10]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
>>> another_sorted_ds = ds.sort(['label', 'text'], reverse=[True, False])
>>> another_sorted_ds['train']['label'][:10]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

shuffledatasets.DatasetDict.shufflehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1211[{"name": "seeds", "val": ": typing.Union[int, dict[str, typing.Optional[int]], NoneType] = None"}, {"name": "seed", "val": ": typing.Optional[int] = None"}, {"name": "generators", "val": ": typing.Optional[dict[str, numpy.random._generator.Generator]] = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "load_from_cache_file", "val": ": typing.Optional[bool] = None"}, {"name": "indices_cache_file_names", "val": ": typing.Optional[dict[str, typing.Optional[str]]] = None"}, {"name": "writer_batch_size", "val": ": typing.Optional[int] = 1000"}]- seeds (Dict[str, int] or int , optional) --
A seed to initialize the default BitGenerator if generator=None .
Nếu Không có thì entropy mới, không thể đoán trước sẽ được lấy khỏi HĐH.
If an int or array_like[ints] is passed, then it will be passed to SeedSequence to derive

trạng thái BitGenerator ban đầu.
Bạn có thể cung cấp một hạt giống cho mỗi tập dữ liệu trong từ điển tập dữ liệu.

- seed (int , optional) --
  A seed to initialize the default BitGenerator if generator=None . Alias for seeds (a ValueError is raised if both are provided).
- generators (Dict[str, *optional*, np.random.Generator]) --
  Trình tạo ngẫu nhiên Numpy được sử dụng để tính toán hoán vị của các hàng tập dữ liệu. If generator=None (default), uses np.random.default_rng (the default BitGenerator (PCG64) of NumPy).
  Bạn phải cung cấp một trình tạo cho mỗi tập dữ liệu trong từ điển tập dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Giữ tập dữ liệu trong bộ nhớ thay vì ghi nó vào tệp bộ đệm.
- load_from_cache_file (Optional[bool] , defaults to True if caching is enabled) --
  Nếu một tệp bộ đệm lưu trữ tính toán hiện tại từ hàm có thể được xác định, hãy sử dụng nó thay vì tính toán lại.
- indices_cache_file_names (Dict[str, str] , optional) --
  Cung cấp tên đường dẫn cho tệp bộ đệm. Nó được sử dụng để lưu trữ các ánh xạ chỉ mục thay vì tên tệp bộ đệm được tạo tự động.
  Bạn phải cung cấp một cache_file_name cho mỗi tập dữ liệu trong từ điển tập dữ liệu.
- writer_batch_size (int , defaults to 1000) --
  Số lượng hàng cho mỗi thao tác ghi đối với trình ghi tệp bộ đệm.
  Giá trị này là sự cân bằng tốt giữa việc sử dụng bộ nhớ trong quá trình xử lý và tốc độ xử lý.
  Giá trị cao hơn khiến quá trình xử lý thực hiện ít tra cứu hơn, giá trị thấp hơn tiêu thụ ít hơn bộ nhớ tạm thời trong khi chạy bản đồ .0
  Tạo Bộ dữ liệu mới trong đó các hàng được xáo trộn.

Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Hiện tại việc xáo trộn sử dụng các trình tạo ngẫu nhiên khó hiểu.
Bạn có thể cung cấp NumPy BitGenerator để sử dụng hoặc hạt giống để bắt đầu mặc định của NumPy random generator (PCG64).

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds["train"]["label"][:10]
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

# đặt hạt giống
>>> shuffled_ds = ds.shuffle(seed=42)
>>> shuffled_ds["train"]["label"][:10]
[0, 1, 0, 1, 0, 0, 0, 0, 0, 0]
```

set_formatdatasets.DatasetDict.set_formathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L575[{"name": "type", "val": ": typing.Optional[str] = None"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}, {"name": "**format_kwargs", "val": ""}]- type (str , optional) -- Loại đầu ra được chọn trong [None, 'numpy', 'torch', 'tensorflow', 'jax', 'arrow', 'pandas', 'polars'] . None means __getitem__ returns python objects (default).

- columns (list[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  None means __getitem__ returns all columns (default).
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects),
- **format_kwargs (additional keyword arguments) --
  Các đối số từ khóa được truyền cho hàm chuyển đổi như np.array , torch.tensor hoặc tensorflow.ragged.constant .0
  Set __getitem__ return format (type and columns).
  Định dạng được đặt cho mọi tập dữ liệu trong từ điển tập dữ liệu.

Có thể gọi bản đồ sau khi gọi set_format. Vì bản đồ có thể thêm các cột mới nên danh sách các cột được định dạng được cập nhật. Trong trường hợp này, nếu bạn áp dụng bản đồ trên tập dữ liệu để thêm cột mới thì điều này cột sẽ được định dạng:

new formatted columns = (all columns - previously unformatted columns)

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> ds = ds.map(lambda x: tokenizer(x["text"], truncation=True, padding=True), batched=True)
>>> ds.set_format(type="numpy", columns=['input_ids', 'token_type_ids', 'attention_mask', 'label']
>>> ds["train"].format
{'columns': ['input_ids', 'token_type_ids', 'attention_mask', 'label'],
 'format_kwargs': {},
 'output_all_columns': Sai,
 'type': 'numpy'}
```

reset_formatdatasets.DatasetDict.reset_formathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L626[]

Đặt lại định dạng trả về __getitem__ cho đối tượng python và tất cả các cột.

Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Same as self.set_format()

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> ds = ds.map(lambda x: tokenizer(x["text"], truncation=True, padding=True), batched=True)
>>> ds.set_format(type="numpy", columns=['input_ids', 'token_type_ids', 'attention_mask', 'label']
>>> ds["train"].format
{'columns': ['input_ids', 'token_type_ids', 'attention_mask', 'label'],
 'format_kwargs': {},
 'output_all_columns': Sai,
 'type': 'numpy'}
>>> ds.reset_format()
>>> ds["train"].format
{'columns': ['text', 'label', 'input_ids', 'token_type_ids', 'attention_mask'],
 'format_kwargs': {},
 'output_all_columns': Sai,
 'type': None}
```

formatted_asdatasets.DatasetDict.formatted_ashttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L535[{"name": "type", "val": ": typing.Optional[str] = None"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}, {"name": "**format_kwargs", "val": ""}]- type (str , optional) --
Loại đầu ra được chọn trong
[None, 'numpy', 'torch', 'tensorflow', 'jax', 'arrow', 'pandas', 'polars'] .
None means __getitem__ returns python objects (default).

- columns (list[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  None means __getitem__ returns all columns (default).
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
- **format_kwargs (additional keyword arguments) --
  Các đối số từ khóa được truyền cho hàm chuyển đổi như np.array , torch.tensor hoặc tensorflow.ragged.constant .0
  To be used in a with statement. Set __getitem__ return format (type and columns).
  Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

with_formatdatasets.DatasetDict.with_formathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L687[{"name": "type", "val": ": typing.Optional[str] = None"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}, {"name": "**format_kwargs", "val": ""}]- type (str , optional) --
Loại đầu ra được chọn trong
[None, 'numpy', 'torch', 'tensorflow', 'jax', 'arrow', 'pandas', 'polars'] .
None means __getitem__ returns python objects (default).

- columns (list[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  None means __getitem__ returns all columns (default).
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
- **format_kwargs (additional keyword arguments) --
  Các đối số từ khóa được truyền cho hàm chuyển đổi như np.array , torch.tensor hoặc tensorflow.ragged.constant .0
  Set __getitem__ return format (type and columns). The data formatting is applied on-the-

bay.

The format type (for example "numpy") is used to format batches when using __getitem__ .

Định dạng được đặt cho mọi tập dữ liệu trong từ điển tập dữ liệu.

It's also possible to use custom transforms for formatting using with_transform().

Contrary to set_format(), with_format returns a new DatasetDict object with new Dataset đồ vật.

Ví dụ:

```
  >>> from datasets import load_dataset
  >>> from transformers import AutoTokenizer
  >>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
  >>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
  >>> ds = ds.map(lambda x: tokenizer(x['text'], truncation=True, padding=True), batched=True)
  >>> ds["train"].format
  {'columns': ['text', 'label', 'input_ids', 'token_type_ids', 'attention_mask'],
    'format_kwargs': {},
    'output_all_columns': Sai,
    'type': None}
  >>> ds = ds.with_format("torch")
  >>> ds["train"].format
  {'columns': ['text', 'label', 'input_ids', 'token_type_ids', 'attention_mask'],
    'format_kwargs': {},
    'output_all_columns': Sai,
    'type': 'torch'}
  >>> ds["train"][0]
  {'text': 'compassionately explores the seemingly irreconcilable situation between conservative chr
    'label': tensor(1),
    'input_ids': tensor([101, 18027, 16310, 16001,1103,9321,178, 11604,7235,6617,
                1742,2165,2820,1206,6588, 22572, 12937,1811,2153,1105,
                1147, 12890, 19587,6463,1105, 15026,1482,119,102,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0,0,0,0,0,0,0,
                    0,0,0,0]),
    'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
    'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])}
```

with_transformdatasets.DatasetDict.with_transformhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L764[{"name": "transform", "val": ":

typing.Optional[typing.Callable]"}, {"name": "columns", "val": ": typing.Optional[list] = None"}, {"name": "output_all_columns", "val": ": bool = False"}]- transform (Callable , optional) --
User-defined formatting transform, replaces the format defined by set_format().
A formatting function is a callable that takes a batch (as a dict) as input and returns a batch.
Hàm này được áp dụng ngay trước khi trả về các đối tượng trong __getitem__ .

- columns (list[str] , optional) --
  Các cột cần định dạng ở đầu ra.
  Nếu được chỉ định thì lô đầu vào của phép biến đổi chỉ chứa các cột đó.
- output_all_columns (bool , defaults to False) --
  Keep un-formatted columns as well in the output (as python objects).
  Nếu được đặt thành True thì các cột chưa được định dạng khác sẽ được giữ nguyên với đầu ra của biến đổi.0
  Đặt định dạng trả về __getitem__ bằng cách sử dụng phép biến đổi này. Việc chuyển đổi được áp dụng n
  các đợt khi __getitem__ được gọi.
  Biến đổi được đặt cho mọi tập dữ liệu trong từ điển tập dữ liệu

As set_format(), this can be reset using reset_format().

Contrary to set_transform() , with_transform returns a new DatasetDict object with new
Các đối tượng tập dữ liệu

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> def encode(example):
...return tokenizer(example['text'], truncation=True, padding=True, return_tensors="pt")
>>> ds = ds.with_transform(encode)
>>> ds["train"][0]
{'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
  1, 1, 1, 1, 1, 1, 1, 1, 1]),
  'input_ids': tensor([101,1103,2067,1110, 17348,1106,1129,1103,6880,1432,
            112,188,1207,107, 14255,1389,107,1105,1115,1119,
            112,188,1280,1106,1294,170, 24194,1256,3407,1190,
            170, 11791,5253,188,1732,7200, 10947, 12606,2895,117,
            179,7766,118,172, 15554,1181,3498,6961,3263,1137,
            188,1566,7912, 14516,6997,119,102]),
  'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0, 0])}
```

Flattendatasets.DatasetDict.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L197[{"name": "max_depth", "val": " = 16"}]

Flatten the Apache Arrow Table of each split (nested features are flatten).

Mỗi cột có kiểu cấu trúc được làm phẳng thành một cột trên mỗi trường cấu trúc.

Các cột khác không thay đổi.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("rajpurkar/squad")
>>> ds["train"].features
{'id': Value('string'),
 'title': Value('string'),
 'context': Value('string'),
 'question': Value('string'),
 'answers.text': List(Value('string')),
 'answers.answer_start': List(Value('int32'))}
>>> ds.flatten()
DatasetDict({
    train: Dataset({
        features: ['id', 'title', 'context', 'question', 'answers.text', 'answers.answer_start'],
        num_rows: 87599
    })
    validation: Dataset({
        features: ['id', 'title', 'context', 'question', 'answers.text', 'answers.answer_start'],
        num_rows: 10570
    })
})
```

castdatasets.DatasetDict.casthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L278[{"name": "features", "val": ": Features"}]- features (Features) --
Các tính năng mới để truyền tập dữ liệu tới.
Tên và thứ tự của các trường trong đối tượng địa lý phải khớp với tên cột hiện tại.
Loại dữ liệu cũng phải có thể chuyển đổi từ loại này sang loại khác.
For non-trivial conversion, e.g. string <-> ClassLabel you should use map() to update the tập dữ liệu.0

Truyền tập dữ liệu sang một bộ tính năng mới.
Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset, ClassLabel, Value
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds["train"].features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
>>> new_features = ds["train"].features.copy()
>>> new_features['label'] = ClassLabel(names=['bad', 'good'])
>>> new_features['text'] = Value('large_string')
>>> ds = ds.cast(new_features)
>>> ds["train"].features
{'label': ClassLabel(names=['bad', 'good']),
 'text': Value('large_string')}
```

cast_columndatasets.DatasetDict.cast_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L310[{"name": "column", "val": ": str"}, {"name": "feature", "val": ""}]- column (str) --
Tên cột.

- feature (Feature) --
  Tính năng mục tiêu.0DatasetDict
  Cột truyền tới tính năng để giải mã.

Ví dụ:

```
>>> from datasets import load_dataset, ClassLabel
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds["train"].features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
>>> ds = ds.cast_column('label', ClassLabel(names=['bad', 'good']))
>>> ds["train"].features
{'label': ClassLabel(names=['bad', 'good']),
 'text': Value('string')}
```

Remove_columnsdatasets.DatasetDict.remove_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L339[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}]- column_names (Union[str, list[str]]) --
Name of the column(s) to remove.0DatasetDictA copy of the dataset object without the

cột cần loại bỏ.

Remove one or several column(s) from each split in the dataset
and the features associated to the column(s).

Việc chuyển đổi được áp dụng cho tất cả các phần tách của từ điển tập dữ liệu.

You can also remove a column using map() with remove_columns but the present method không sao chép dữ liệu của các cột còn lại và do đó nhanh hơn.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds = ds.remove_columns("label")
DatasetDict({
    train: Dataset({
        features: ['text'],
        num_rows: 8530
    })
    validation: Dataset({
        features: ['text'],
        num_rows: 1066
    })
    test: Dataset({
        features: ['text'],
        num_rows: 1066
    })
})
```

đổi tên_columndatasets.DatasetDict.rename_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L381[{"name": "original_column_name", "val": ": str"}, {"name": "new_column_name", "val": ": str"}]- original_column_name (str) --
Tên cột cần đổi tên.

- new_column_name (str) --
  Tên mới cho cột.0

Đổi tên một cột trong tập dữ liệu và di chuyển các tính năng được liên kết với cột ban đầu

dưới tên cột mới.
Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

You can also rename a column using map() with remove_columns but the present method:

- đảm nhiệm việc di chuyển các đối tượng ban đầu dưới tên cột mới.
- không sao chép dữ liệu sang tập dữ liệu mới và do đó nhanh hơn nhiều.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds = ds.rename_column("label", "label_new")
DatasetDict({
    train: Dataset({
        features: ['text', 'label_new'],
        num_rows: 8530
    })
    validation: Dataset({
        features: ['text', 'label_new'],
        num_rows: 1066
    })
    test: Dataset({
        features: ['text', 'label_new'],
        num_rows: 1066
    })
})
```

đổi tên_columnsdatasets.DatasetDict.rename_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L429[{"name": "column_mapping", "val": ": dict"}]- column_mapping (Dict[str, str]) --
Ánh xạ các cột để đổi tên thành tên mới.0DatasetDictMột bản sao của tập dữ liệu với cột được đổi tên.

Đổi tên một số cột trong tập dữ liệu và di chuyển các tính năng được liên kết với bản gốc cột bên dưới
tên cột mới.
Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.rename_columns({'text': 'text_new', 'label': 'label_new'})
DatasetDict({
    train: Dataset({
        features: ['text_new', 'label_new'],
        num_rows: 8530
    })
    validation: Dataset({
        features: ['text_new', 'label_new'],
        num_rows: 1066
    })
    test: Dataset({
        features: ['text_new', 'label_new'],
        num_rows: 1066
    })
})
```

select_columnsdatasets.DatasetDict.select_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L467[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}]- column_names (Union[str, list[str]]) -- Name of the column(s) to keep.0
Select one or several column(s) from each split in the dataset and the features associated to the column(s).

Việc chuyển đổi được áp dụng cho tất cả các phần tách của tập dữ liệu từ điển.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes")
>>> ds.select_columns("text")
DatasetDict({
    train: Dataset({
        features: ['text'],
        num_rows: 8530
    })
    validation: Dataset({
        features: ['text'],
        num_rows: 1066
    })
    test: Dataset({
        features: ['text'],
        num_rows: 1066
    })
})
```

class_encode_columndatasets.DatasetDict.class_encode_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L503[{"name": "column", "val": ": str"}, {"name": "include_nulls", "val": ": bool = False"}]- column (str) --
Tên của cột cần truyền.

- include_nulls (bool , defaults to False) --
  Có bao gồm các giá trị null trong nhãn lớp hay không. Nếu True, giá trị null sẽ được mã hóa làm nhãn lớp "Không".
  0
  Truyền cột đã cho dưới dạng ClassLabel và cập nhật các bảng.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("boolq")
>>> ds["train"].features
{'answer': Value('bool'),
 'passage': Value('string'),
 'question': Value('string')}
>>> ds = ds.class_encode_column("answer")
>>> ds["train"].features
{'answer': ClassLabel(num_classes=2, names=['False', 'True']),
 'passage': Value('string'),
 'question': Value('string')}
```

push_to_hubdatasets.DatasetDict.push_to_hubhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1616[{"name": "repo_id", "val": ""}, {"name": "config_name", "val": ": str = 'default'"}, {"name": "set_default", "val": ": typing.Optional[bool] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "commit_message", "val": ": typing.Optional[str] = None"}, {"name": "commit_description", "val": ": typing.Optional[str] = None"}, {"name": "private", "val": ": typing.Optional[bool] = None"}, {"name": "token", "val": ": typing.Optional[str] = None"}, {"name": "revision", "val": ": typing.Optional[str] = None"}, {"name": "create_pr", "val": ": typing.Optional[bool] = False"}, {"name": "max_shard_size", "val": ": typing.Union[str, int, NoneType] = None"}, {"name": "num_shards", "val": ": typing.Optional[dict[str, int]] = None"}, {"name": "embed_external_files", "val": ": bool = True"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}]- repo_id (str) --
The ID of the repository to push to in the following format: <user>/<dataset_name> or <org>/<dataset_name> . Also accepts <dataset_name> , which will default to the namespace của người dùng đã đăng nhập.

- config_name (str) --
  Tên cấu hình của tập dữ liệu. Mặc định là "mặc định".
- set_default (bool , optional) --
  Có đặt cấu hình này làm cấu hình mặc định hay không. Ngược lại, cấu hình mặc định là cái một
  được đặt tên là "mặc định".
- data_dir (str , optional) --
  Tên thư mục sẽ chứa các tệp dữ liệu được tải lên. Mặc định là config_name nếu

khác biệt
từ "mặc định", "dữ liệu" khác.

• commit_message (str , optional) --
Thông báo để cam kết trong khi đẩy. Sẽ mặc định là "Tải lên tập dữ liệu".

• commit_description (str , optional) --
Mô tả cam kết sẽ được tạo.
Additionally, description of the PR if a PR is created (create_pr is True).

• private (bool , optional) --
Whether to make the repo private. If None (default), the repo will be public unless the
mặc định của tổ chức là riêng tư. Giá trị này bị bỏ qua nếu repo đã tồn tại.

• token (str , optional) --
Mã thông báo xác thực tùy chọn cho Hugging Face Hub. Nếu không có mã thông báo nào được chuyển, mặc định
vào mã thông báo được lưu cục bộ khi đăng nhập bằng deathface-cli login . Sẽ gây ra lỗi
nếu không có mã thông báo nào được chuyển và người dùng chưa đăng nhập.

• revision (str , optional) --
Branch để đẩy các tập tin đã tải lên. Mặc định là nhánh "chính".

• create_pr (bool , optional, defaults to False) --
Có nên tạo PR bằng các tệp đã tải lên hay cam kết trực tiếp.

• max_shard_size (int or str , optional, defaults to "500MB") --
Kích thước tối đa của các phân đoạn dữ liệu sẽ được tải lên trung tâm. Nếu được biểu thị dưới dạng chuỗi, cần phải là chữ số theo sau là đơn vị
(like "500MB" or "1GB").

• num_shards (Dict[str, int] , optional) --
Số lượng mảnh để viết. Theo mặc định, số lượng phân đoạn phụ thuộc vào max_shard_size .
Sử dụng từ điển để xác định num_shards khác nhau cho mỗi phần tách.

• embed_external_files (bool , defaults to True) --
Có nhúng byte tệp vào phân đoạn hay không.
Cụ thể, điều này sẽ thực hiện những việc sau trước khi đẩy các trường thuộc loại:
Âm thanh và Hình ảnh xóa thông tin đường dẫn cục bộ và nhúng nội dung tệp vào
Tập tin sàn gỗ.

• num_proc (int , optional, defaults to None) --
Số lượng quy trình khi chuẩn bị và tải lên tập dữ liệu.
Điều này hữu ích nếu tập dữ liệu được tạo từ nhiều mẫu hoặc tệp phương tiện để nhúng.
Đa xử lý bị tắt theo mặc định.

0huggingface_hub.CommitInfo
Đẩy DatasetDict vào trung tâm dưới dạng tập dữ liệu Parquet.
DatasetDict được đẩy bằng các yêu cầu HTTP và không cần phải có git hoặc
git-lfs đã được cài đặt.

Mỗi phần chia tập dữ liệu sẽ được đẩy độc lập. Tập dữ liệu được đẩy sẽ giữ nguyên phần phân chia ban đầu những cái tên.

Theo mặc định, các tệp Parquet kết quả được tự chứa: nếu tập dữ liệu của bạn chứa Hình ảnh hoặc Âm thanh
data, tệp Parquet sẽ lưu trữ byte hình ảnh hoặc tệp âm thanh của bạn.
Bạn có thể tắt tính năng này bằng cách đặt embed_external_files thành Sai.

Ví dụ:

```
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>")
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>", private=True)
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>", max_shard_size="1GB")
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>", num_shards={"train": 1024, "test": 8})
```

If you want to add a new configuration (or subset) to a dataset (e.g. if the dataset has multiple tasks/versions/languages):

```
>>> english_dataset.push_to_hub("<organization>/<dataset_id>", "en")
>>> french_dataset.push_to_hub("<organization>/<dataset_id>", "fr")
>>> # later
>>> english_dataset = load_dataset("<organization>/<dataset_id>", "en")
>>> french_dataset = load_dataset("<organization>/<dataset_id>", "fr")
```

save_to_diskdatasets.DatasetDict.save_to_diskhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1294[{"name": "dataset_dict_path", "val": ": typing.Union[str, bytes, os.PathLike]"}, {"name": "max_shard_size", "val": ": typing.Union[str, int, NoneType] = None"}, {"name": "num_shards", "val": ": typing.Optional[dict[str, int]] = None"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}]- dataset_dict_path (path-like) -- Path (e.g. dataset/train) or remote URI (e.g. s3://my-bucket/dataset/train)
của thư mục dict tập dữ liệu nơi tập dữ liệu dict sẽ được lưu vào.

- max_shard_size (int or str , optional, defaults to "500MB") --
  Kích thước tối đa của các phân đoạn dữ liệu sẽ được lưu vào hệ thống tệp. Nếu được biểu thị dưới dạng chuỗi, cần phải là chữ số theo sau là đơn vị
  (like "50MB").
- num_shards (Dict[str, int] , optional) --
  Số lượng mảnh để viết. Theo mặc định, số lượng phân đoạn phụ thuộc vào max_shard_size và num_proc.
  Bạn cần cung cấp số lượng phân đoạn cho mỗi tập dữ liệu trong từ điển tập dữ liệu.
  Sử dụng từ điển để xác định num_shards khác nhau cho mỗi phần tách.
- num_proc (int , optional, default None) --
  Số lượng quy trình khi tải xuống và tạo tập dữ liệu cục bộ.
  Đa xử lý bị tắt theo mặc định.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0

Lưu một tập dữ liệu vào hệ thống tập tin bằng cách sử dụng fsspec.spec.AbstractFileSystem .

Đối với dữ liệu Hình ảnh, Âm thanh và Video:

All the Image(), Audio() and Video() data are stored in the arrow files.
If you want to store paths or urls, please use the Value("string") type.

Ví dụ:

```
>>> dataset_dict.save_to_disk("path/to/dataset/directory")
>>> dataset_dict.save_to_disk("path/to/dataset/directory", max_shard_size="1GB")
>>> dataset_dict.save_to_disk("path/to/dataset/directory", num_shards={"train": 1024, "test": 8})
```

Load_from_diskdatasets.DatasetDict.load_from_diskhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1368[{"name": "dataset_dict_path", "val": ": typing.Union[str, bytes, os.PathLike]"}, {"name": "keep_in_memory", "val": ": typing.Optional[bool] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}]- dataset_dict_path (path-like) --
Path (e.g. "dataset/train") or remote URI (e.g. "s3//my-bucket/dataset/train")
của thư mục dict tập dữ liệu nơi tập dữ liệu dict sẽ được tải từ đó.

- keep_in_memory (bool , defaults to None) --
  Có sao chép tập dữ liệu vào bộ nhớ hay không. Nếu Không , thì
  tập dữ liệu sẽ không được sao chép trong bộ nhớ trừ khi được bật rõ ràng bằng cách cài đặt
  bộ dữ liệu.config.IN_MEMORY_MAX_SIZE thành khác không. Xem thêm chi tiết tại
  phần cải thiện hiệu suất.
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
  0Bộ dữ liệuDict

Tải tập dữ liệu đã được lưu trước đó bằng save_to_disk từ hệ thống tệp bằng cách sử dụng fsspec.spec.AbstractFileSystem .

Ví dụ:

```
>>> ds = load_from_disk('path/to/dataset/directory')
```

from_csvdatasets.DatasetDict.from_csvhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1428[{"name": "path_or_paths", "val": ": dict"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "**kwargs", "val": ""}]- path_or_paths (dict of path-like) --
Path(s) of the CSV file(s).

- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str, optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển tới pandas.read_csv .0DatasetDict
  Create DatasetDict from CSV file(s).

Ví dụ:

```
>>> from datasets import DatasetDict
>>> ds = DatasetDict.from_csv({'train': 'path/to/dataset.csv'})
```

from_jsondatasets.DatasetDict.from_jsonhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1471[{"name": "path_or_paths", "val": ": dict"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "**kwargs", "val": ""}]- path_or_paths (path-like or list of path-like) --
Path(s) of the JSON Lines file(s).

- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str, optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển tới JsonConfig .0DatasetDict
  Create DatasetDict from JSON Lines file(s).

Ví dụ:

```
>>> from datasets import DatasetDict
>>> ds = DatasetDict.from_json({'train': 'path/to/dataset.json'})
```

from_parquetdatasets.DatasetDict.from_parquethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1514[{"name": "path_or_paths", "val": ": dict"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "columns", "val": ": typing.Optional[list[str]] = None"}, {"name": "**kwargs", "val": ""}]- path_or_paths (dict of path-like) --
Path(s) of the CSV file(s).

- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str , optional, defaults to "~/.cache/huggingface/datasets") --

Thư mục để lưu trữ dữ liệu.

- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- columns (list[str] , optional) --
  Nếu không phải None , chỉ những cột này sẽ được đọc từ tệp.
  Tên cột có thể là tiền tố của trường lồng nhau, ví dụ: 'a' sẽ chọn
  'a.b', 'a.c' và 'a.d.e'.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển tới ParquetConfig .0DatasetDict
  Create DatasetDict from Parquet file(s).

Ví dụ:

```
>>> from datasets import DatasetDict
>>> ds = DatasetDict.from_parquet({'train': 'path/to/dataset/parquet'})
```

from_textdatasets.DatasetDict.from_texthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L1563[{"name": "path_or_paths", "val": ": dict"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "cache_dir", "val": ": str = None"}, {"name": "keep_in_memory", "val": ": bool = False"}, {"name": "**kwargs", "val": ""}]- path_or_paths (dict of path-like) --
Path(s) of the text file(s).

- features (Features, optional) --
  Tính năng của bộ dữ liệu.
- cache_dir (str , optional, defaults to "~/.cache/huggingface/datasets") --
  Thư mục để lưu trữ dữ liệu.
- keep_in_memory (bool , defaults to False) --
  Có sao chép dữ liệu trong bộ nhớ hay không.
- **kwargs (additional keyword arguments) --
  Đối số từ khóa được chuyển tới TextConfig .0DatasetDict
  Create DatasetDict from text file(s).

Ví dụ:

```
>>> from datasets import DatasetDict
>>> ds = DatasetDict.from_text({'train': 'path/to/dataset.txt'})
```

IterableDatasetdatasets.IterableDataset

Lớp cơ sở IterableDataset triển khai Bộ dữ liệu có thể lặp lại được hỗ trợ bởi các trình tạo python.

bộ dữ liệu lớp.IterableDatasetdatasets.IterableDatasethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2125[{"name": "ex_iterable", "val": ": _BaseExamplesIterable"}, {"name": "info", "val": ": typing.Optional[datasets.info.DatasetInfo] = None"}, {"name": "split", "val": ": typing.Optional[datasets.splits.NamedSplit] = None"}, {"name": "formatting", "val": ": typing.Optional[datasets.iterable_dataset.FormattingConfig] = None"}, {"name": "shuffling", "val": ": typing.Optional[datasets.iterable_dataset.ShufflingConfig] = None"}, {"name": "distributed", "val": ": typing.Optional[datasets.iterable_dataset.DistributedConfig] = None"}, {"name": "token_per_repo_id", "val": ": typing.Optional[dict[str, typing.Union[str, bool, NoneType]]] = None"}]
Bộ dữ liệu được hỗ trợ bởi một lần lặp.

from_generatordatasets.IterableDataset.from_generatorhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2522[{"name": "generator", "val": ": typing.Callable"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "gen_kwargs", "val": ": typing.Optional[dict] = None"}, {"name": "split", "val": ": NamedSplit = NamedSplit('train')"}]- generator (Callable) --
Một hàm tạo tạo ra các ví dụ.

- features (Features , optional) --
  Tính năng của bộ dữ liệu.
- gen_kwargs(dict , optional) --
  Đối số từ khóa được chuyển đến trình tạo có thể gọi được.
  Bạn có thể xác định một tập dữ liệu có thể lặp lại được phân đoạn bằng cách chuyển danh sách các phâ
  Điều này có thể được sử dụng để cải thiện việc xáo trộn và khi lặp lại tập dữ liệu với nhiều công nhân.
- split (NamedSplit, defaults to Split.TRAIN) --

Tên phân chia được gán cho tập dữ liệu.

0 Bộ dữ liệu có thể lặp lại

Tạo một Bộ dữ liệu có thể lặp lại từ một trình tạo.

Ví dụ:

```
>>> def gen():
...yield {"text": "Good", "label": 0}
...yield {"text": "Bad", "label": 1}
None
>>> ds = IterableDataset.from_generator(gen)
```

```
>>> 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
```

Remove_columnsdatasets.IterableDataset.remove_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3254[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}]- column_names (Union[str, List[str]]) --
Name of the column(s) to remove.0 IterableDataset A copy of the dataset object without the cột cần loại bỏ.

Remove one or several column(s) in the dataset and the features associated to them.
Việc xóa được thực hiện nhanh chóng trên các ví dụ khi lặp qua tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> next(iter(ds))
{'text': 'the rock is destined to be the 21st century's new " conan " and that he's going to make
>>> ds = ds.remove_columns("label")
>>> next(iter(ds))
{'text': 'the rock is destined to be the 21st century's new " conan " and that he's going to make
```

select_columnsdatasets.IterableDataset.select_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3289[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}]- column_names (Union[str, List[str]]) --
Name of the column(s) to select.0 IterableDataset A copy of the dataset object with selected cột.
Select one or several column(s) in the dataset and the features
liên quan đến họ. Việc lựa chọn được thực hiện nhanh chóng trên các ví dụ
khi lặp qua tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> next(iter(ds))
{'text': 'the rock is destined to be the 21st century's new " conan " and that he's going to make
>>> ds = ds.select_columns("text")
>>> next(iter(ds))
{'text': 'the rock is destined to be the 21st century's new " conan " and that he's going to make
```

cast_columndatasets.IterableDataset.cast_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3340[{"name": "column", "val": ": str"}, {"name": "feature", "val": ": typing.Union[dict, list, tuple, datasets.features.features.Value,
bộ dữ liệu.features.features.ClassLabel, bộ dữ liệu.features.translation.Translation,
bộ dữ liệu.features.translation.TranslationVariableLanguages,
bộ dữ liệu.features.features.LargeList, bộ dữ liệu.features.features.List,
bộ dữ liệu.features.features.Array2D, bộ dữ liệu.features.features.Array3D,
bộ dữ liệu.features.features.Array4D, bộ dữ liệu.features.features.Array5D,
bộ dữ liệu.features.audio.Audio, bộ dữ liệu.features.image.Image, bộ dữ liệu.features.video.Video,

datasets.features.pdf.Pdf]"}]- column (str) --
Tên cột.

- feature (Feature) --
  Tính năng mục tiêu.0 IterableDataset
  Cột truyền tới tính năng để giải mã.

Ví dụ:

```
>>> from datasets import load_dataset, Audio
>>> ds = load_dataset("PolyAI/minds14", name="en-US", split="train", streaming=True)
>>> ds.features
{'audio': Audio(sampling_rate=8000, mono=True, decode=True, id=None),
 'english_transcription': Value('string'),
 'intent_class': ClassLabel(num_classes=14, names=['abroad', 'address', 'app_error', 'atm_limit',
 'lang_id': ClassLabel(num_classes=14, names=['cs-CZ', 'de-DE', 'en-AU', 'en-GB', 'en-US', 'es-ES'
 'path': Value('string'),
 'transcription': Value('string')}
>>> ds = ds.cast_column("audio", Audio(sampling_rate=16000))
>>> ds.features
{'audio': Audio(sampling_rate=16000, mono=True, decode=True, id=None),
 'english_transcription': Value('string'),
 'intent_class': ClassLabel(num_classes=14, names=['abroad', 'address', 'app_error', 'atm_limit',
 'lang_id': ClassLabel(num_classes=14, names=['cs-CZ', 'de-DE', 'en-AU', 'en-GB', 'en-US', 'es-ES'
 'path': Value('string'),
 'transcription': Value('string')}
```

castdatasets.IterableDataset.casthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3387[{"name": "features", "val": ": Features"}]- features (Features) --
Các tính năng mới để truyền tập dữ liệu tới.
Tên của các trường trong đối tượng địa lý phải khớp với tên cột hiện tại.
Loại dữ liệu cũng phải có thể chuyển đổi từ loại này sang loại khác.
For non-trivial conversion, e.g. string <-> ClassLabel you should use map() to update the Dataset.0 IterableDataset Một bản sao của tập dữ liệu với các tính năng được truyền.

Truyền tập dữ liệu sang một bộ tính năng mới.

Ví dụ:

```
>>> from datasets import load_dataset, ClassLabel, Value
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> ds.features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
>>> new_features = ds.features.copy()
>>> new_features["label"] = ClassLabel(names=["bad", "good"])
>>> new_features["text"] = Value("large_string")
>>> ds = ds.cast(new_features)
>>> ds.features
{'label': ClassLabel(names=['bad', 'good']),
 'text': Value('large_string')}
```

giải mãdatasets.IterableDataset.decodehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3434[{"name": "enable", "val": ": bool = True"}, {"name": "num_threads", "val": ": int = 0"}]- enable (bool , defaults to True) --
Bật hoặc tắt tính năng giải mã.

- num_threads (int , defaults to 0) --
  Bật đa luồng để giải mã các tính năng.0 IterableDataset Một bản sao của tập dữ liệu với các tính năng đúc.

Bật hoặc tắt tính năng giải mã tập dữ liệu cho âm thanh, hình ảnh, video.

When enabled (default), media types are decoded:

- audio -> dict of "array" and "sampling_rate" and "path"
- image -> PIL.Image
- video -> torchvision.io.VideoReader

Bạn có thể kích hoạt đa luồng bằng cách sử dụng num_threads. Điều này đặc biệt hữu ích để tăng tốc xa
data streaming. However it can be slower than num_threads=0 for local data on fast disks.

Việc tắt giải mã rất hữu ích nếu bạn muốn lặp lại đường dẫn hoặc byte của tệp phương tiện without actually decoding their content. To disable decoding you can use .decode(False) , cái mà

is equivalent to calling .cast() or .cast_column() with all the Audio, Image and Video types set to decode=False .

Ví dụ:

Tắt giải mã:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("sshh12/planet-textures", split="train", streaming=True)
>>> next(iter(ds))
{'image': <PIL.PngImagePlugin.PngImageFile image mode=RGB size=2048x1024>,
'văn bản': 'Một thiên thể ở xa có lớp vỏ băng giá, hiển thị màu xanh nhạt, được bao phủ bởi
>>> ds = ds.decode(False)
>>> ds.features
{'image': Image(mode=None, decode=False, id=None),
'text': Value('string')}
>>> next(iter(ds))
{
    'image': {
        'đường dẫn': 'hf://datasets/sshh12/planet-textures@69dc4cef7a5c4b2cfe387727ec8ea73d4bff7302/tra
        'byte': Không có
    },
    'văn bản': 'Một thiên thể ở xa có lớp vỏ băng giá, hiển thị màu xanh nhạt, được bao phủ bởi
}
```

Tăng tốc độ phát trực tuyến với đa luồng:

```
>>> import os
>>> from datasets import load_dataset
>>> from tqdm import tqdm
>>> ds = load_dataset("sshh12/planet-textures", split="train", streaming=True)
>>> num_threads = min(32, (os.cpu_count() or 1) + 4)
>>> ds = ds.decode(num_threads=num_threads)
>>> for _ in tqdm(ds):# 20 times faster !
None
```

iterdatasets.IterableDataset.iterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2459[]

iterdatasets.IterableDataset.iterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2484[{"name": "batch_size", "val": ": int"}, {"name": "drop_last_batch", "val": ": bool = False"}]- batch_size (int) -- size of each batch to yield.

- drop_last_batch (bool , default False) -- Whether a last batch smaller than the batch_size nên là đánh rơi0 Lặp lại qua các lô có kích thước batch_size.

mapdatasets.IterableDataset.maphttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2694[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "with_indices", "val": ": bool = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "drop_last_batch", "val": ": bool = False"}, {"name": "remove_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "features", "val": ": typing.Optional[datasets.features.features.Features] = None"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}]- function (Callable , optional, defaults to None) -- Hàm được áp dụng nhanh chóng trên các ví dụ khi bạn lặp lại trên tập dữ liệu. Nó phải có một trong các chữ ký sau:

- function(example: Dict[str, Any]) -> Dict[str, Any] if batched=False and with_indices=False
- function(example: Dict[str, Any], idx: int) -> Dict[str, Any] if batched=False and with_indices=True
- function(batch: Dict[str, List]) -> Dict[str, List] if batched=True and with_indices=False
- function(batch: Dict[str, List], indices: List[int]) -> Dict[str, List] if batched=True and with_indices=True

Để sử dụng nâng cao, hàm cũng có thể trả về pyarrow.Table . Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song. Moreover if your function returns nothing (None), then map will run your function and return tập dữ liệu không thay đổi Nếu không có chức năng nào được cung cấp, mặc định là chức năng nhận dạng: lambda x: x .

- with_indices (bool , defaults to False) --

Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này chữ ký của hàm
should be def function(example, idx[, rank]): ... .

- input_columns (Optional[Union[str, List[str]]] , defaults to None) --
  Các cột được truyền vào hàm
  như các đối số vị trí. Nếu Không có, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được chuyển dư
  một lý lẽ.
- batched (bool , defaults to False) --
  Cung cấp hàng loạt ví dụ để hoạt động.
- batch_size (int , optional, defaults to 1000) --
  Number of examples per batch provided to function if batched=True .
  batch_size <= 0 or batch_size == None then provide the full dataset as a single batch to
  chức năng .
- drop_last_batch (bool , defaults to False) --
  Liệu lô cuối cùng có nhỏ hơn batch_size hay không
  bị loại bỏ thay vì được xử lý bởi hàm.
- remove_columns ([List[str]] , optional, defaults to None) --
  Xóa vùng chọn cột trong khi thực hiện ánh xạ.
  Các cột sẽ bị xóa trước khi cập nhật các ví dụ với đầu ra của hàm , tức là nếu
  chức năng đang thêm
  các cột có tên trong Remove_columns , các cột này sẽ được giữ lại.
- features ([Features] , optional, defaults to None) --
  Các loại tính năng của tập dữ liệu kết quả.
- fn_kwargs (Dict , optional, default None) --
  Đối số từ khóa được truyền cho hàm .0

Apply a function to all the examples in the iterable dataset (individually or in batches) and
cập nhật chúng.
Nếu hàm của bạn trả về một cột đã tồn tại thì nó sẽ ghi đè lên cột đó.
Hàm này được áp dụng nhanh chóng trên các ví dụ khi lặp qua tập dữ liệu.

Bạn có thể chỉ định xem hàm có nên được bó hay không bằng tham số được bó:

- Nếu bó là Sai thì hàm sẽ lấy 1 mẫu và trả về 1 mẫu.
  An example is a dictionary, e.g. {"text": "Hello there !"} .
- Nếu batched là True và batch_size là 1 thì hàm sẽ lấy một batch gồm 1 ví dụ như
  đầu vào và có thể trả về một lô có 1 hoặc nhiều mẫu.

A batch is a dictionary, e.g. a batch of 1 example is {"text": ["Hello there !"]}.

- If batched is True and batch_size is n > 1, then the function takes a batch of n ví dụ làm đầu vào và có thể trả về một lô có n ví dụ hoặc với số lượng tùy ý ví dụ.
  Lưu ý rằng đợt cuối cùng có thể có ít hơn n mẫu.
  A batch is a dictionary, e.g. a batch of n examples is {"text": ["Hello there !"] * n} .

Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song, với tối đa một hàng ngàn cuộc gọi mô phỏng.
Bạn nên sử dụng asyncio.Semaphore trong hàm của mình nếu bạn muốn đặt mức tối đa số thao tác có thể thực hiện đồng thời.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> def add_prefix(example):
...example["text"] = "Review: " + example["text"]
...trả lại ví dụ
>>> ds = ds.map(add_prefix)
>>> list(ds.take(3))
[{'label': 1,
  'text': 'Đánh giá: tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ đi
  {'label': 1,
  'text': 'Đánh giá: phần tiếp theo được xây dựng công phu tuyệt đẹp của bộ ba phim "chúa tể của những ch
  {'label': 1, 'text': 'Review: effective but too-tepid biopic'}]
```

đổi tên_columndatasets.IterableDataset.rename_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3199[{"name": "original_column_name", "val": ": str"}, {"name": "new_column_name", "val": ": str"}]- original_column_name (str) -- Tên cột cần đổi tên.

- new_column_name (str) --
  Tên mới cho cột.0 IterableDataset Một bản sao của tập dữ liệu có tên được đổi tên cột.

Đổi tên một cột trong tập dữ liệu và di chuyển các tính năng được liên kết với cột ban đầu dưới cột mới

tên.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> next(iter(ds))
{'label': 1,
 'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
>>> ds = ds.rename_column("text", "movie_review")
>>> next(iter(ds))
{'label': 1,
 'movie_review': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ đi
```

filterdatasets.IterableDataset.filterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2846[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "with_indices", "val": " = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}]- function (Callable) --
Có thể gọi được với một trong các chữ ký sau:

- function(example: Dict[str, Any]) -> bool if with_indices=False, batched=False
- function(example: Dict[str, Any], indices: int) -> bool if with_indices=True, batched=False
- function(example: Dict[str, List]) -> List[bool] if with_indices=False, batched=True
- function(example: Dict[str, List], indices: List[int]) -> List[bool] if with_indices=True, batched=True

Nếu hàm không đồng bộ thì bộ lọc sẽ chạy song song hàm của bạn.
Nếu không có hàm nào được cung cấp, mặc định là hàm luôn True: lambda x: True .

- with_indices (bool , defaults to False) --
  Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này chữ ký của hàm should be def function(example, idx): ... .
- input_columns (str or List[str] , optional) --
  Các cột được chuyển vào chức năng như

những lập luận mang tính vị trí. Nếu Không, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được ch
lý lẽ.

- batched (bool , defaults to False) --
  Cung cấp hàng loạt ví dụ để hoạt động.
- batch_size (int , optional, default 1000) --
  Number of examples per batch provided to function if batched=True .
- fn_kwargs (Dict , optional, default None) --
  Đối số từ khóa được truyền cho hàm .0
  Áp dụng chức năng lọc cho tất cả các phần tử để tập dữ liệu chỉ bao gồm các ví dụ
  theo chức năng lọc.
  Việc lọc được thực hiện nhanh chóng khi lặp qua tập dữ liệu.

Nếu hàm không đồng bộ thì bộ lọc sẽ chạy hàm của bạn song song, với tối đa một
thousand simulatenous calls (configurable).
Bạn nên sử dụng asyncio.Semaphore trong hàm của mình nếu bạn muốn đặt mức tối đa
số thao tác có thể thực hiện đồng thời.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> ds = ds.filter(lambda x: x["label"] == 0)
>>> list(ds.take(3))
[{'label': 0, 'movie_review': 'simplistic , silly and tedious .'},
 {'label': 0,
 'movie_review': "nó quá trẻ trung và trẻ con , chỉ có những cậu thiếu niên mới có thể thấy nó buồn cười ."
 {'label': 0,
 'movie_review': 'bóc lột và hầu như không có chiều sâu hay sự tinh tế có thể khiến w
```

shuffledatasets.IterableDataset.shufflehttps://github.com/huggingface/datasets/blob/4.2.0/src/
datasets/iterable_dataset.py#L2932[{"name": "seed", "val": " = None"}, {"name": "generator",
"val": ": typing.Optional[numpy.random._generator.Generator] = None"}, {"name": "buffer_size",
"val": ": int = 1000"}]- seed (int , optional, defaults to None) --
Hạt giống ngẫu nhiên sẽ được sử dụng để xáo trộn tập dữ liệu.
Nó được sử dụng để lấy mẫu từ bộ đệm xáo trộn và cũng để xáo trộn các phân đoạn dữ liệu.

- generator (numpy.random.Generator , optional) --

Trình tạo ngẫu nhiên Numpy được sử dụng để tính toán hoán vị của các hàng tập dữ liệu.
If generator=None (default), uses np.random.default_rng (the default BitGenerator (PCG64) of NumPy).

- buffer_size (int , defaults to 1000) --
  Kích thước của bộ đệm.0

Xáo trộn ngẫu nhiên các phần tử của tập dữ liệu này.

Tập dữ liệu này lấp đầy bộ đệm bằng các phần tử buffer_size, sau đó lấy mẫu ngẫu nhiên các phần tử từ bộ đệm này,
thay thế các phần tử đã chọn bằng các phần tử mới. Để xáo trộn hoàn hảo, kích thước bộ đệm lớn hơn hơn hoặc
bằng với kích thước đầy đủ của tập dữ liệu là bắt buộc.

Ví dụ: nếu tập dữ liệu của bạn chứa 10.000 phần tử nhưng buffer_size được đặt thành 1000 thì xáo trộn ý chí
ban đầu chọn một phần tử ngẫu nhiên chỉ từ 1000 phần tử đầu tiên trong bộ đệm. Một lần phần tử là
selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element,
duy trì bộ đệm 1000 phần tử.

Nếu tập dữ liệu được tạo thành từ nhiều phân đoạn, nó cũng sẽ xáo trộn thứ tự của các phân đoạn.
However if the order has been fixed by using skip() or take()
thì thứ tự của các mảnh được giữ không thay đổi.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> list(ds.take(3))
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'}]
>>> shuffled_ds = ds.shuffle(seed=42)
>>> list(shuffled_ds.take(3))
[{'label': 1,
  'text': "một bộ phim thể thao với những pha hành động hấp dẫn trên sân và một câu chuyện mà bạn quan
  {'label': 1,
  'text': 'at its best , the good girl is a refreshingly adult take on adultery . . .'},
  {'label': 1,
  'text': "sam jones đã trở thành một nhà làm phim rất may mắn vào ngày Wilco bị loại khỏi hãng thu âm củ
```

batchdatasets.IterableDataset.batchhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3558[{"name": "batch_size", "val": ": int"}, {"name": "drop_last_batch", "val": ": bool = False"}]- batch_size (int) -- The number of samples in mỗi đợt.

- drop_last_batch (bool , defaults to False) -- Whether to drop the last incomplete lô.0

Nhóm các mẫu từ tập dữ liệu thành các đợt.

Ví dụ:

```
>>> ds = load_dataset("some_dataset", streaming=True)
>>> batched_ds = ds.batch(batch_size=32)
```

Skipdatasets.IterableDataset.skiphttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3006[{"name": "n", "val": ": int"}]- n (int) -- Số phần tử cần bỏ qua.0

Tạo một IterableDataset mới bỏ qua n phần tử đầu tiên.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> list(ds.take(3))
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'}]
>>> ds = ds.skip(1)
>>> list(ds.take(3))
[{'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'},
  {'label': 1,
  'text': 'nếu thỉnh thoảng bạn thích đi xem phim để giải trí , wasabi là nơi tốt để bắt đầu
```

takedatasets.IterableDataset.takehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3093[{"name": "n", "val": ": int"}]- n (int) --
Số phần tử cần lấy.0

Tạo một IterableDataset mới chỉ với n phần tử đầu tiên.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train", streaming=True)
>>> small_ds = ds.take(2)
>>> list(small_ds)
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
```

sharddatasets.IterableDataset.shardhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3130[{"name": "num_shards", "val": ": int"}, {"name": "index", "val": ": int"}, {"name": "contiguous", "val": ": bool = True"}]- num_shards (int) --

Có bao nhiêu phân đoạn để chia tập dữ liệu thành.

- index (int) --
  Phân đoạn nào để chọn và trả lại.
- contiguous -- (bool , defaults to True):
  Có nên chọn các khối chỉ mục liền kề cho phân đoạn hay không.0
  Trả về phân đoạn chỉ mục -nth từ tập dữ liệu được chia thành các phần num_shards.

This shards deterministically. dataset.shard(n, i) splits the dataset into contiguous chunks, nên có thể dễ dàng nối lại với nhau sau khi xử lý. Nếu như
dataset.num_shards % n == l , then the
first l datasets each have (dataset.num_shards // n) + 1 shards, and the remaining datasets have (dataset.num_shards // n) shards.
datasets.concatenate_datasets([dset.shard(n, i) for i in range(n)]) returns a dataset with thứ tự giống như ban đầu.
In particular, dataset.shard(dataset.num_shards, i) returns a dataset with 1 shard.

Lưu ý: n phải nhỏ hơn hoặc bằng số lượng phân đoạn trong tập dữ liệu.num_shards.

On the other hand, dataset.shard(n, i, contiguous=False) contains all the shards of the dataset whose index mod n = i .

Be sure to shard before using any randomizing operator (such as shuffle).
Tốt nhất là nên sử dụng toán tử phân đoạn sớm trong quy trình dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("amazon_polarity", split="train", streaming=True)
>>> ds
Dataset({
    features: ['label', 'title', 'content'],
    num_shard: 4
})
>>> ds.shard(num_shards=2, index=0)
Dataset({
    features: ['label', 'title', 'content'],
    num_shard: 2
})
```

lặp lạidatasets.IterableDataset.repeathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3050[{"name": "num_times", "val": ": typing.Optional[int]"}]-
num_times (int) or (None) --
Số lần lặp lại tập dữ liệu. Nếu Không, tập dữ liệu sẽ được lặp lại vô thời hạn.0

Tạo một IterableDataset mới lặp lại tập dữ liệu cơ bản num_times lần.

N.B. Hiệu quả của việc gọi ngẫu nhiên sau khi lặp lại phụ thuộc đáng kể vào kích thước bộ đệm.
Với buffer_size 1, dữ liệu trùng lặp sẽ không bao giờ được nhìn thấy trong cùng một lần lặp, ngay cả sau khi
ds.repeat(n).shuffle(seed=42, buffer_size=1) is equivalent to ds.shuffle(seed=42,
buffer_size=1).repeat(n),
và chỉ xáo trộn các thứ tự phân đoạn trong mỗi lần lặp.
With buffer size >= (num samples in the dataset * num_times), we get full shuffling of the
dữ liệu lặp lại, tức là chúng ta có thể quan sát các bản sao trong
cùng một lần lặp.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> ds = ds.take(2).repeat(2)
>>> list(ds)
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'},
  {'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'}]
```

to_csvdatasets.IterableDataset.to_csvhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3688[{"name": "path_or_buf", "val": ": typing.Union[str, bytes, os.PathLike, typing.BinaryIO]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**to_csv_kwargs", "val": ""}]- path_or_buf (PathLike or FileOrBuffer) --
Either a path to a file (e.g. file.csv), a remote URI (e.g. hf://datasets/username/my_dataset_name/data.csv), hoặc BinaryIO, nơi tập dữ liệu sẽ được lưu ở định dạng đã chỉ định.

- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
- **to_csv_kwargs (additional keyword arguments) --
  Các tham số cần truyền tới pandas.DataFrame.to_csv của gấu trúc.
  Chỉ số tham số mặc định là Sai nếu không được chỉ định.
  If you would like to write the index, pass index=True and also set a name for the index cột theo
  truyền index_label .0 int Số lượng ký tự hoặc byte được ghi.
  Xuất tập dữ liệu sang csv.

Việc này lặp lại trên tập dữ liệu và tải nó hoàn toàn vào bộ nhớ trước khi ghi.

Ví dụ:

```
>>> ds.to_csv("path/to/dataset/directory")
```

to_pandasdatasets.IterableDataset.to_pandashttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3622[{"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "batched", "val": ": bool = False"}]- batch_size (int , optional) --
The size (number of rows) of the batches if batched is True .
Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .

- batched (bool) --
  Đặt thành True để trả về trình tạo tạo ra tập dữ liệu dưới dạng lô
  of batch_size rows. Defaults to False (returns the whole datasets
  once).0 pandas.DataFrame or Iterator[pandas.DataFrame]
  Trả về tập dữ liệu dưới dạng pandas.DataFrame . Cũng có thể trả lại một máy phát điện lớn bộ dữ liệu.

Ví dụ:

```
>>> ds.to_pandas()
```

to_dictdatasets.IterableDataset.to_dicthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3584[{"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "batched", "val": ": bool = False"}]- batch_size (int , optional) -- The size (number of rows) of the batches if batched is True .
Defaults to datasets.config.DEFAULT_MAX_BATCH_SIZE .0 dict or Iterator[dict]
Trả về tập dữ liệu dưới dạng lệnh Python. Cũng có thể trả về một trình tạo cho các tập dữ liệu lớn.

Ví dụ:

```
>>> ds.to_dict()
```

to_jsondatasets.IterableDataset.to_jsonhttps://github.com/huggingface/datasets/blob/4.2.0/src/

datasets/iterable_dataset.py#L3731[{"name": "path_or_buf", "val": ": typing.Union[str, bytes, os.PathLike, typing.BinaryIO]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**to_json_kwargs", "val": ""}]- path_or_buf (PathLike or FileOrBuffer) --
Either a path to a file (e.g. file.json), a remote URI (e.g. hf://datasets/username/my_dataset_name/data.json),
hoặc BinaryIO, nơi tập dữ liệu sẽ được lưu ở định dạng đã chỉ định.

- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
- **to_json_kwargs (additional keyword arguments) --
  Các tham số cần truyền tới pandas.DataFrame.to_json của gấu trúc.
  Default arguments are lines=True and
  orient="records". The parameter index defaults to False if orient is "split" or "table
  " . If you would like to write the index, pass index=Tru
  e .</paramsdesc><paramgroups>0</paramgroups><rettype> int`The number of characters or byte được viết.
  Xuất tập dữ liệu sang Dòng JSON hoặc JSON.

Việc này lặp lại trên tập dữ liệu và tải nó hoàn toàn vào bộ nhớ trước khi ghi.

Định dạng đầu ra mặc định là Dòng JSON.
To export to JSON, pass lines=False argument and the desired orient .

Ví dụ:

```
>>> ds.to_json("path/to/dataset/directory/filename.jsonl")
```

```
>>> num_shards = dataset.num_shards
>>> for index in range(num_shards):
...shard = dataset.shard(index, num_shards)
...shard.to_json(f"path/of/my/dataset/data-{index:05d}.jsonl")
```

to_parquetdatasets.IterableDataset.to_parquethttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3827[{"name": "path_or_buf", "val": ": typing.Union[str, bytes, os.PathLike, typing.BinaryIO]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "storage_options", "val": ": typing.Optional[dict] = None"}, {"name": "**parquet_writer_kwargs", "val": ""}]- path_or_buf (PathLike or FileOrBuffer) --
Either a path to a file (e.g. file.parquet), a remote URI (e.g. hf://datasets/username/my_dataset_name/data.parquet),
hoặc BinaryIO, nơi tập dữ liệu sẽ được lưu ở định dạng đã chỉ định.

- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .
- storage_options (dict , optional) --
  Các cặp khóa/giá trị sẽ được chuyển đến phần phụ trợ hệ thống tệp, nếu có.
- **parquet_writer_kwargs (additional keyword arguments) --
  Các tham số cần truyền tới pyarrow.parquet.ParquetWriter .0 int của PyArrow Số lượng ký tự hoặc byte được viết.
  Xuất tập dữ liệu sang sàn gỗ

Ví dụ:

```
>>> ds.to_parquet("path/to/dataset/directory")
```

```
>>> num_shards = dataset.num_shards
>>> for index in range(num_shards):
...shard = dataset.shard(index, num_shards)
...shard.to_parquet(f"path/of/my/dataset/data-{index:05d}.parquet")
```

to_sqldatasets.IterableDataset.to_sqlhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L3785[{"name": "name", "val": ": str"}, {"name": "con", "val": ": typing.Union[str, ForwardRef('sqlalchemy.engine.Connection'), ForwardRef('sqlalchemy.engine.Engine'), ForwardRef('sqlite3.Connection')]"}, {"name": "batch_size", "val": ": typing.Optional[int] = None"}, {"name": "**sql_writer_kwargs", "val": ""}]- name (str) --
Tên bảng SQL.

- con (str or sqlite3.Connection or sqlalchemy.engine.Connection or sqlalchemy.engine.Connection) --
  Chuỗi URI hoặc đối tượng kết nối SQLite3/SQLAlchemy được sử dụng để ghi vào cơ sở dữ liệu.
- batch_size (int , optional) --
  Kích thước của lô để tải vào bộ nhớ và ghi cùng một lúc.
  Mặc định làdatadatas.config.DEFAULT_MAX_BATCH_SIZE .
- **sql_writer_kwargs (additional keyword arguments) --
  Các tham số cần truyền tới pandas.DataFrame.to_sql của gấu trúc.
  Chỉ số tham số mặc định là Sai nếu không được chỉ định.
  If you would like to write the index, pass index=True and also set a name for the index cột theo
  truyền index_label .0 int Số lượng bản ghi được ghi.
  Xuất tập dữ liệu sang cơ sở dữ liệu SQL.

Ví dụ:

```
>>> # con provided as a connection URI string
>>> ds.to_sql("data", "sqlite:///my_own_db.sql")
>>> # con provided as a sqlite3 connection object
>>> import sqlite3
>>> con = sqlite3.connect("my_own_db.sql")
>>> with con:
...ds.to_sql("data", con)
```

push_to_hubdatasets.IterableDataset.push_to_hubhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L4032[{"name": "repo_id", "val": ": str"}, {"name": "config_name", "val": ": str = 'default'"}, {"name": "set_default", "val": ": typing.Optional[bool] = None"}, {"name": "split", "val": ": typing.Optional[str] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "commit_message", "val": ": typing.Optional[str] = None"}, {"name": "commit_description", "val": ": typing.Optional[str] = None"}, {"name": "private", "val": ": typing.Optional[bool] = None"}, {"name": "token", "val": ": typing.Optional[str] = None"}, {"name": "revision", "val": ": typing.Optional[str] = None"}, {"name": "create_pr", "val": ": typing.Optional[bool] = False"}, {"name": "num_shards", "val": ": typing.Optional[int] = None"}, {"name": "embed_external_files", "val": ": bool = True"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}]- repo_id (str) --
The ID of the repository to push to in the following format: <user>/<dataset_name> or

<org>/<dataset_name> . Also accepts <dataset_name> , which will default to the namespace của người dùng đã đăng nhập.

- config_name (str , defaults to "default") --
  The configuration name (or subset) of a dataset. Defaults to "default".
- set_default (bool , optional) --
  Có đặt cấu hình này làm cấu hình mặc định hay không. Ngược lại, cấu hình mặc định là cái một
  được đặt tên là "mặc định".
- split (str , optional) --
  Tên của phần tách sẽ được đặt cho tập dữ liệu đó. Mặc định là self.split .
- data_dir (str , optional) --
  Tên thư mục sẽ chứa các tệp dữ liệu được tải lên. Mặc định là config_name nếu khác biệt
  từ "mặc định", "dữ liệu" khác.
- commit_message (str , optional) --
  Thông báo để cam kết trong khi đẩy. Sẽ mặc định là "Tải lên tập dữ liệu".
- commit_description (str , optional) --
  Mô tả cam kết sẽ được tạo.
  Additionally, description of the PR if a PR is created (create_pr is True).
- private (bool , optional) --
  Whether to make the repo private. If None (default), the repo will be public unless the
  mặc định của tổ chức là riêng tư. Giá trị này bị bỏ qua nếu repo đã tồn tại.
- token (str , optional) --
  Mã thông báo xác thực tùy chọn cho Hugging Face Hub. Nếu không có mã thông báo nào được chuyển, mặc định
  vào mã thông báo được lưu cục bộ khi đăng nhập bằng deathface-cli login . Sẽ gây ra lỗi
  nếu không có mã thông báo nào được chuyển và người dùng chưa đăng nhập.
- revision (str , optional) --
  Branch để đẩy các tập tin đã tải lên. Mặc định là nhánh "chính".
- create_pr (bool , optional, defaults to False) --
  Có nên tạo PR bằng các tệp đã tải lên hay cam kết trực tiếp.
- num_shards (int , optional) --
  Số lượng mảnh để viết. Theo mặc định, bằng .num_shards của tập dữ liệu này.
- embed_external_files (bool , defaults to True) --

Có nhúng byte tệp vào phân đoạn hay không.

Cụ thể, điều này sẽ thực hiện những việc sau trước khi đẩy các trường thuộc loại:

Âm thanh và Hình ảnh: xóa thông tin đường dẫn cục bộ và nhúng nội dung tệp vào Tập tin sàn gỗ.

- num_proc (int , optional, defaults to None) --

Số lượng quy trình khi chuẩn bị và tải lên tập dữ liệu.

Điều này rất hữu ích nếu tập dữ liệu được tạo thành từ nhiều mẫu và các phép biến đổi.

Đa xử lý bị tắt theo mặc định.0huggingface_hub.CommitInfo

Đẩy tập dữ liệu vào trung tâm dưới dạng tập dữ liệu Parquet.

Tập dữ liệu được đẩy bằng các yêu cầu HTTP và không cần phải có git hoặc git-lfs được cài đặt.

Theo mặc định, các tệp Parquet kết quả được tự chứa. Nếu tập dữ liệu của bạn chứa Hình ảnh, Âm thanh hoặc Video
dữ liệu, tệp Parquet sẽ lưu trữ byte hình ảnh hoặc tệp âm thanh của bạn.
Bạn có thể tắt tính năng này bằng cách đặt embed_external_files thành False .

Ví dụ:

```
>>> dataset.push_to_hub("<organization>/<dataset_id>")
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>", private=True)
>>> dataset.push_to_hub("<organization>/<dataset_id>", num_shards=1024)
```

If your dataset has multiple splits (e.g. train/validation/test):

```
>>> train_dataset.push_to_hub("<organization>/<dataset_id>", split="train")
>>> val_dataset.push_to_hub("<organization>/<dataset_id>", split="validation")
>>> # later
>>> dataset = load_dataset("<organization>/<dataset_id>")
>>> train_dataset = dataset["train"]
>>> val_dataset = dataset["validation"]
```

If you want to add a new configuration (or subset) to a dataset (e.g. if the dataset has multiple tasks/versions/languages):

```
>>> english_dataset.push_to_hub("<organization>/<dataset_id>", "en")
>>> french_dataset.push_to_hub("<organization>/<dataset_id>", "fr")
>>> # later
>>> english_dataset = load_dataset("<organization>/<dataset_id>", "en")
>>> french_dataset = load_dataset("<organization>/<dataset_id>", "fr")
```

Load_state_dictdatasets.IterableDataset.load_state_dicthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2242[{"name": "state_dict", "val": ": dict"}]
Tải state_dict của tập dữ liệu.
Việc lặp lại sẽ khởi động lại ở ví dụ tiếp theo kể từ khi trạng thái được lưu.

Việc tiếp tục trả về chính xác nơi điểm kiểm tra đã được lưu ngoại trừ hai trường hợp:

1. các ví dụ từ bộ đệm xáo trộn bị mất khi tiếp tục và bộ đệm được nạp lại bằng bộ đệm mới dữ liệu
2. combinations of .with_format(arrow) and batched .map() may skip one batch.

Ví dụ:

```
>>> from datasets import Dataset, concatenate_datasets
>>> ds = Dataset.from_dict({"a": range(6)}).to_iterable_dataset(num_shards=3)
>>> for idx, example in enumerate(ds):
...print(example)
...if idx == 2:
...state_dict = ds.state_dict()
...print("checkpoint")
...phá vỡ
>>> ds.load_state_dict(state_dict)
>>> print(f"restart from checkpoint")
>>> for example in ds:
...print(example)
```

trả về:

{'a': 0}
{'a': 1}
{'a': 2}
trạm kiểm soát
khởi động lại từ điểm kiểm tra
{'a': 3}
{'a': 4}
{'a': 5}

```
>>> from torchdata.stateful_dataloader import StatefulDataLoader
>>> ds = load_dataset("deepmind/code_contests", streaming=True, split="train")
>>> dataloader = StatefulDataLoader(ds, batch_size=32, num_workers=4)
>>> # checkpoint
>>> state_dict = dataloader.state_dict()# uses ds.state_dict() under the hood
>>> # resume from checkpoint
>>> dataloader.load_state_dict(state_dict)# uses ds.load_state_dict() under the hood
```

state_dictdatasets.IterableDataset.state_dicthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2189[] dict
Lấy state_dict hiện tại của tập dữ liệu.
Nó tương ứng với trạng thái ở ví dụ mới nhất mà nó mang lại.

Việc tiếp tục trả về chính xác nơi điểm kiểm tra đã được lưu ngoại trừ hai trường hợp:

1. các ví dụ từ bộ đệm xáo trộn bị mất khi tiếp tục và bộ đệm được nạp lại bằng bộ đệm mới dữ liệu
2. combinations of .with_format(arrow) and batched .map() may skip one batch.

Ví dụ:

```
>>> from datasets import Dataset, concatenate_datasets
>>> ds = Dataset.from_dict({"a": range(6)}).to_iterable_dataset(num_shards=3)
>>> for idx, example in enumerate(ds):
...print(example)
...if idx == 2:
...state_dict = ds.state_dict()
...print("checkpoint")
...phá vỡ
>>> ds.load_state_dict(state_dict)
>>> print(f"restart from checkpoint")
>>> for example in ds:
...print(example)
```

trả về:

```
{'a': 0}
{'a': 1}
{'a': 2}
trạm kiểm soát
khởi động lại từ điểm kiểm tra
{'a': 3}
{'a': 4}
{'a': 5}
```

```
>>> from torchdata.stateful_dataloader import StatefulDataLoader
>>> ds = load_dataset("deepmind/code_contests", streaming=True, split="train")
>>> dataloader = StatefulDataLoader(ds, batch_size=32, num_workers=4)
>>> # checkpoint
>>> state_dict = dataloader.state_dict()# uses ds.state_dict() under the hood
>>> # resume from checkpoint
>>> dataloader.load_state_dict(state_dict)# uses ds.load_state_dict() under the hood
```

infodatasets.IterableDataset.infohttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L167[]
Đối tượng DatasetInfo chứa tất cả siêu dữ liệu trong tập dữ liệu.

Splitdatasets.IterableDataset.splithttps://github.com/huggingface/datasets/blob/4.2.0/src/

datasets/arrow_dataset.py#L172[]
Đối tượng NamedSplit tương ứng với phần tách tập dữ liệu được đặt tên.

builder_namedatasets.IterableDataset.builder_namehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L177[]

trích dẫndatasets.IterableDataset.citehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L181[]

config_namedatasets.IterableDataset.config_namehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L185[]

tập dữ liệu_sizeatasets.IterableDataset.dataset_sizehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L189[]

descriptiondatasets.IterableDataset.descriptionhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L193[]

download_checksumsdatasets.IterableDataset.download_checksumshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L197[]

download_sizeatasets.IterableDataset.download_sizehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L201[]

featuredatasets.IterableDataset.featureshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L205[]

trang chủdatasets.IterableDataset.homepagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L209[]

licensedatasets.IterableDataset.licensehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L213[]

size_in_bytesdatasets.IterableDataset.size_in_byteshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L217[]

giám sát_keysdatasets.IterableDataset.supervised_keyshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/arrow_dataset.py#L221[]

versiondatasets.IterableDataset.versionhttps://github.com/huggingface/datasets/blob/4.2.0/src/

datasets/arrow_dataset.py#L225[]

bộ dữ liệu lớp.IterableColumndatasets.IterableColumnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/iterable_dataset.py#L2092[{"name": "source", "val": ": typing.Union[ForwardRef('IterableDataset'), ForwardRef('IterableColumn')]"}, {"name": "column_name", "val": ": str"}]

Một lần lặp cho một cột cụ thể của IterableDataset.

Ví dụ:

Lặp lại các văn bản của cột "văn bản" của tập dữ liệu:

```
for text in dataset["text"]:
    None
```

Nó cũng hoạt động với các cột lồng nhau:

```
for source in dataset["metadata"]["source"]:
    None
```

IterableDatasetDictdatasets.IterableDatasetDict

Dictionary with split names as keys ('train', 'test' for example), and IterableDataset objects as các giá trị.

bộ dữ liệu lớp.IterableDatasetDictdatasets.IterableDatasetDicthttps://github.com/huggingface/bộ dữ liệu/blob/4.2.0/src/datasets/dataset_dict.py#L1986""

mapdatasets.IterableDatasetDict.maphttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2087[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "with_indices", "val": ": bool = False"}, {"name": "with_split", "val": ": bool = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": int = 1000"}, {"name": "drop_last_batch", "val": ": bool = False"}, {"name": "remove_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}]- function (Callable , optional, defaults to None) --

Hàm được áp dụng nhanh chóng trên các ví dụ khi bạn lặp lại trên tập dữ liệu.
Nó phải có một trong các chữ ký sau:

- function(example: Dict[str, Any]) -> Dict[str, Any] if batched=False and with_indices=False
- function(example: Dict[str, Any], idx: int) -> Dict[str, Any] if batched=False and with_indices=True
- function(batch: Dict[str, list]) -> Dict[str, list] if batched=True and with_indices=False
- function(batch: Dict[str, list], indices: list[int]) -> Dict[str, list] if batched=True and with_indices=True

Để sử dụng nâng cao, hàm cũng có thể trả về pyarrow.Table .
Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song.
Moreover if your function returns nothing (None), then map will run your function and return
tập dữ liệu không thay đổi
Nếu không có chức năng nào được cung cấp, mặc định là chức năng nhận dạng: lambda x: x .

- with_indices (bool , defaults to False) --
  Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này chữ ký của hàm
  should be def function(example, idx[, rank]): ... .
- input_columns ([Union[str, list[str]]] , optional, defaults to None) --
  Các cột được truyền vào hàm
  như các đối số vị trí. Nếu Không có, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được chuyển dư
  một lý lẽ.
- batched (bool , defaults to False) --
  Cung cấp hàng loạt ví dụ để hoạt động.
- batch_size (int , optional, defaults to 1000) --
  Number of examples per batch provided to function if batched=True .
- drop_last_batch (bool , defaults to False) --
  Liệu lô cuối cùng có nhỏ hơn batch_size hay không
  bị loại bỏ thay vì được xử lý bởi hàm.
- remove_columns ([list[str]] , optional, defaults to None) --
  Xóa vùng chọn cột trong khi thực hiện ánh xạ.
  Các cột sẽ bị xóa trước khi cập nhật các ví dụ với đầu ra của hàm , tức là nếu
  chức năng đang thêm

các cột có tên trong Remove_columns , các cột này sẽ được giữ lại.

- fn_kwargs (Dict , optional, defaults to None) --
  Đối số từ khóa được chuyển đến hàm 0

Apply a function to all the examples in the iterable dataset (individually or in batches) and cập nhật chúng.
Nếu hàm của bạn trả về một cột đã tồn tại thì nó sẽ ghi đè lên cột đó.
Hàm này được áp dụng nhanh chóng trên các ví dụ khi lặp qua tập dữ liệu.
Việc chuyển đổi được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Bạn có thể chỉ định xem hàm có nên được bó hay không bằng tham số được bó:

- Nếu bó là Sai thì hàm sẽ lấy 1 mẫu và trả về 1 mẫu.
  An example is a dictionary, e.g. {"text": "Hello there !"} .
- Nếu batched là True và batch_size là 1 thì hàm sẽ lấy một batch gồm 1 ví dụ như đầu vào và có thể trả về một lô có 1 hoặc nhiều mẫu.
  A batch is a dictionary, e.g. a batch of 1 example is {"text": ["Hello there !"]} .
- If batched is True and batch_size is n > 1, then the function takes a batch of n ví dụ làm đầu vào và có thể trả về một lô có n ví dụ hoặc với số lượng tùy ý ví dụ.
  Lưu ý rằng đợt cuối cùng có thể có ít hơn n mẫu.
  A batch is a dictionary, e.g. a batch of n examples is {"text": ["Hello there !"] * n} .

Nếu hàm không đồng bộ thì bản đồ sẽ chạy hàm của bạn song song, với tối đa một nghìn cuộc gọi đồng thời.
Bạn nên sử dụng asyncio.Semaphore trong hàm của mình nếu bạn muốn đặt mức tối đa số thao tác có thể thực hiện đồng thời.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> def add_prefix(example):
...example["text"] = "Review: " + example["text"]
...trả lại ví dụ
>>> ds = ds.map(add_prefix)
>>> next(iter(ds["train"]))
{'label': 1,
  'text': 'Đánh giá: tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ đi
```

filterdatasets.IterableDatasetDict.filterhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2187[{"name": "function", "val": ": typing.Optional[typing.Callable] = None"}, {"name": "with_indices", "val": " = False"}, {"name": "input_columns", "val": ": typing.Union[str, list[str], NoneType] = None"}, {"name": "batched", "val": ": bool = False"}, {"name": "batch_size", "val": ": typing.Optional[int] = 1000"}, {"name": "fn_kwargs", "val": ": typing.Optional[dict] = None"}]- function (Callable) --
Có thể gọi được với một trong các chữ ký sau:

- function(example: Dict[str, Any]) -> bool if with_indices=False, batched=False
- function(example: Dict[str, Any], indices: int) -> bool if with_indices=True, batched=False
- function(example: Dict[str, list]) -> list[bool] if with_indices=False, batched=True
- function(example: Dict[str, list], indices: list[int]) -> list[bool] if with_indices=True, batched=True

Nếu không có hàm nào được cung cấp, mặc định là hàm luôn True: lambda x: True .

- with_indices (bool , defaults to False) --
  Cung cấp các chỉ số mẫu để hoạt động. Lưu ý rằng trong trường hợp này chữ ký của hàm should be def function(example, idx): ... .
- input_columns (str or list[str] , optional) --
  Các cột được chuyển vào chức năng như những lập luận mang tính vị trí. Nếu Không, ánh xạ chính tả tới tất cả các cột được định dạng sẽ được ch
  lý lẽ.
- batched (bool , defaults to False) --
  Cung cấp hàng loạt ví dụ để hoạt động
- batch_size (int , optional, defaults to 1000) --

Number of examples per batch provided to function if batched=True .

- fn_kwargs (Dict , optional, defaults to None) --

  Đối số từ khóa được chuyển đến hàm 0

  Áp dụng chức năng lọc cho tất cả các phần tử để tập dữ liệu chỉ bao gồm các ví dụ theo chức năng lọc.

  Việc lọc được thực hiện nhanh chóng khi lặp qua tập dữ liệu.

  Việc lọc được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> ds = ds.filter(lambda x: x["label"] == 0)
>>> list(ds["train"].take(3))
[{'label': 0, 'text': 'Review: simplistic , silly and tedious .'},
 {'label': 0,
 'text': "Đánh giá: nó quá trẻ trung và trẻ con , chỉ có những cậu thiếu niên mới có thể thấy nó buồn cười ."
 {'label': 0,
 'text': 'Đánh giá: bóc lột và hầu như không có chiều sâu hoặc sự tinh vi có thể khiến w
```

shuffledatasets.IterableDatasetDict.shufflehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2250[{"name": "seed", "val": " = None"}, {"name": "generator", "val": ": typing.Optional[numpy.random._generator.Generator] = None"}, {"name": "buffer_size", "val": ": int = 1000"}]- seed (int , optional, defaults to None) --
Hạt giống ngẫu nhiên sẽ được sử dụng để xáo trộn tập dữ liệu.
Nó được sử dụng để lấy mẫu từ bộ đệm xáo trộn và cũng để xáo trộn các phân đoạn dữ liệu.

- generator (numpy.random.Generator , optional) --

  Trình tạo ngẫu nhiên Numpy được sử dụng để tính toán hoán vị của các hàng tập dữ liệu.

  If generator=None (default), uses np.random.default_rng (the default BitGenerator (PCG64) of NumPy).

- buffer_size (int , defaults to 1000) --

  Kích thước của bộ đệm.0

Xáo trộn ngẫu nhiên các phần tử của tập dữ liệu này.
Việc xáo trộn được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Tập dữ liệu này lấp đầy bộ đệm với các phần tử buffer_size, sau đó lấy mẫu ngẫu nhiên các phần tử từ đây đệm,
thay thế các phần tử đã chọn bằng các phần tử mới. Để xáo trộn hoàn hảo, kích thước bộ đệm lớn hơn hơn hoặc
bằng với kích thước đầy đủ của tập dữ liệu là bắt buộc.

Ví dụ: nếu tập dữ liệu của bạn chứa 10.000 phần tử nhưng buffer_size được đặt thành 1000 thì xáo trộn ý chí
ban đầu chọn một phần tử ngẫu nhiên chỉ từ 1000 phần tử đầu tiên trong bộ đệm. Một lần phần tử là
selected, its space in the buffer is replaced by the next (i.e. 1,001-st) element,
duy trì bộ đệm 1000 phần tử.

Nếu tập dữ liệu được tạo thành từ nhiều phân đoạn, nó cũng sẽ xáo trộn thứ tự của các phân đoạn.
However if the order has been fixed by using skip() or take()
thì thứ tự của các mảnh được giữ không thay đổi.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> list(ds["train"].take(3))
[{'label': 1,
  'text': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ tạo ra
  {'label': 1,
  'text': 'phần tiếp theo được xây dựng công phu và lộng lẫy của bộ ba phim "chúa tể của những chiếc nhẫn
  {'label': 1, 'text': 'effective but too-tepid biopic'}]
>>> ds = ds.shuffle(seed=42)
>>> list(ds["train"].take(3))
[{'label': 1,
  'text': "một bộ phim thể thao với những pha hành động hấp dẫn trên sân và một câu chuyện mà bạn quan
  {'label': 1,
  'text': 'at its best , the good girl is a refreshingly adult take on adultery . . .'},
  {'label': 1,
  'text': "sam jones đã trở thành một nhà làm phim rất may mắn vào ngày Wilco bị loại khỏi hãng thu âm củ
```

with_formatdatasets.IterableDatasetDict.with_formathttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2041[{"name": "type", "val": ": typing.Optional[str] =

None"}]- type (str , optional) --

Loại đầu ra được chọn trong

[None, 'numpy', 'torch', 'tensorflow', 'jax', 'arrow', 'pandas', 'polars'] .

None means it returns python objects (default).0

Trả về một tập dữ liệu có định dạng được chỉ định.

Ví dụ:

```
>>> from datasets import load_dataset
>>> from transformers import AutoTokenizer
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="validation", streaming=T
>>> tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
>>> ds = ds.map(lambda x: tokenizer(x['text'], truncation=True, padding=True), batched=True)
>>> ds = ds.with_format("torch")
>>> next(iter(ds))
{'text': 'compassionately explores the seemingly irreconcilable situation between conservative chr
 'label': tensor(1),
 'input_ids': tensor([101, 18027, 16310, 16001,1103,9321,178, 11604,7235,6617,
          1742,2165,2820,1206,6588, 22572, 12937,1811,2153,1105,
          1147, 12890, 19587,6463,1105, 15026,1482,119,102,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0,0,0,0,0,0,0,
              0,0,0,0]),
 'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
          0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])}
```

castdatasets.IterableDatasetDict.casthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2458[{"name": "features", "val": ": Features"}]- features (Features)

None

Các tính năng mới để truyền tập dữ liệu tới.
Tên của các trường trong đối tượng phải khớp với tên cột hiện tại.
Loại dữ liệu cũng phải có thể chuyển đổi từ loại này sang loại khác.
For non-trivial conversion, e.g. string <-> ClassLabel you should use map to update the
Dataset.0IterableDatasetDictMột bản sao của tập dữ liệu với các tính năng được truyền.

Truyền tập dữ liệu sang một bộ tính năng mới.
Việc truyền kiểu được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> ds["train"].features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
>>> new_features = ds["train"].features.copy()
>>> new_features['label'] = ClassLabel(names=['bad', 'good'])
>>> new_features['text'] = Value('large_string')
>>> ds = ds.cast(new_features)
>>> ds["train"].features
{'label': ClassLabel(names=['bad', 'good']),
 'text': Value('large_string')}
```

cast_columndatasets.IterableDatasetDict.cast_columnhttps://github.com/huggingface/
datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2427[{"name": "column", "val": ": str"},
{"name": "feature", "val": ": typing.Union[dict, list, tuple, datasets.features.features.Value,
bộ dữ liệu.features.features.ClassLabel, bộ dữ liệu.features.translation.Translation,
bộ dữ liệu.features.translation.TranslationVariableLanguages,
bộ dữ liệu.features.features.LargeList, bộ dữ liệu.features.features.List,
bộ dữ liệu.features.features.Array2D, bộ dữ liệu.features.features.Array3D,
bộ dữ liệu.features.features.Array4D, bộ dữ liệu.features.features.Array5D,
bộ dữ liệu.features.audio.Audio, bộ dữ liệu.features.image.Image, bộ dữ liệu.features.video.Video,
datasets.features.pdf.Pdf]"}]- column (str) --
Tên cột.

- feature (Feature) --

Tính năng mục tiêu.0IterableDatasetDict

Cột truyền tới tính năng để giải mã.

Việc truyền kiểu được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset, ClassLabel
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> ds["train"].features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
>>> ds = ds.cast_column('label', ClassLabel(names=['bad', 'good']))
>>> ds["train"].features
{'label': ClassLabel(names=['bad', 'good']),
 'text': Value('string')}
```

Remove_columnsdatasets.IterableDatasetDict.remove_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2375[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}]- column_names (Union[str, list[str]]) --
Name of the column(s) to remove.0IterableDatasetDictA copy of the dataset object without the cột cần loại bỏ.

Remove one or several column(s) in the dataset and the features associated to them.
Việc xóa được thực hiện nhanh chóng trên các ví dụ khi lặp qua tập dữ liệu.
Việc xóa được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> ds = ds.remove_columns("label")
>>> next(iter(ds["train"]))
{'text': 'the rock is destined to be the 21st century's new " conan " and that he's going to make
```

đổi tên_columndatasets.IterableDatasetDict.rename_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2311[{"name": "original_column_name", "val": ": str"}, {"name": "new_column_name", "val": ": str"}]- original_column_name (str) --

Tên cột cần đổi tên.

- new_column_name (str) --
  Tên mới cho cột.0IterableDatasetDictMột bản sao của tập dữ liệu có tên được đổi tên cột.

Đổi tên một cột trong tập dữ liệu và di chuyển các tính năng được liên kết với cột ban đầu dưới cột mới
tên.
Việc đổi tên được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> ds = ds.rename_column("text", "movie_review")
>>> next(iter(ds["train"]))
{'label': 1,
 'movie_review': 'tảng đá được mệnh danh là " conan " mới của thế kỷ 21 và anh ấy sẽ đi
```

đổi tên_columnsdatasets.IterableDatasetDict.rename_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2347[{"name": "column_mapping", "val": ": dict"}]- column_mapping (Dict[str, str]) --
Ánh xạ các cột để đổi tên thành tên mới của chúng.0IterableDatasetDictMột bản sao của tập dữ liệu với các cột được đổi tên

Đổi tên một số cột trong tập dữ liệu và di chuyển các tính năng được liên kết với bản gốc cột bên dưới
tên cột mới.
Việc đổi tên được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> ds = ds.rename_columns({"text": "movie_review", "label": "rating"})
>>> next(iter(ds["train"]))
{'movie_review': 'the rock is destined to be the 21st century's new " conan " and that he's going
 'rating': 1}
```

select_columnsdatasets.IterableDatasetDict.select_columnshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2401[{"name": "column_names", "val": ": typing.Union[str, list[str]]"}]- column_names (Union[str, list[str]]) --
Name of the column(s) to keep.0IterableDatasetDictA copy of the dataset object with only các cột đã chọn.
Select one or several column(s) in the dataset and the features liên quan đến họ. Việc lựa chọn được thực hiện nhanh chóng trên các ví dụ khi lặp qua tập dữ liệu. Việc lựa chọn được áp dụng cho tất cả các tập dữ liệu của từ điển tập dữ liệu.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", streaming=True)
>>> ds = ds.select("text")
>>> next(iter(ds["train"]))
{'text': 'the rock is destined to be the 21st century's new " conan " and that he's going to make
```

push_to_hubdatasets.IterableDatasetDict.push_to_hubhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/dataset_dict.py#L2495[{"name": "repo_id", "val": ""}, {"name": "config_name", "val": ": str = 'default'"}, {"name": "set_default", "val": ": typing.Optional[bool] = None"}, {"name": "data_dir", "val": ": typing.Optional[str] = None"}, {"name": "commit_message", "val": ": typing.Optional[str] = None"}, {"name": "commit_description", "val": ": typing.Optional[str] = None"}, {"name": "private", "val": ": typing.Optional[bool] = None"}, {"name": "token", "val": ": typing.Optional[str] = None"}, {"name": "revision", "val": ": typing.Optional[str] = None"}, {"name": "create_pr", "val": ": typing.Optional[bool] = False"}, {"name": "num_shards", "val": ": typing.Optional[dict[str, int]] = None"}, {"name": "embed_external_files", "val": ": bool = True"}, {"name": "num_proc", "val": ": typing.Optional[int] = None"}]- repo_id (str) --

The ID of the repository to push to in the following format: <user>/<dataset_name> or <org>/<dataset_name> . Also accepts <dataset_name> , which will default to the namespace của người dùng đã đăng nhập.

- config_name (str) --
  Tên cấu hình của tập dữ liệu. Mặc định là "mặc định".
- set_default (bool , optional) --
  Có đặt cấu hình này làm cấu hình mặc định hay không. Ngược lại, cấu hình mặc định là cái một
  được đặt tên là "mặc định".
- data_dir (str , optional) --
  Tên thư mục sẽ chứa các tệp dữ liệu được tải lên. Mặc định là config_name nếu khác biệt
  từ "mặc định", "dữ liệu" khác.
- commit_message (str , optional) --
  Thông báo để cam kết trong khi đẩy. Sẽ mặc định là "Tải lên tập dữ liệu".
- commit_description (str , optional) --
  Mô tả cam kết sẽ được tạo.
  Additionally, description of the PR if a PR is created (create_pr is True).
- private (bool , optional) --
  Whether to make the repo private. If None (default), the repo will be public unless the
  mặc định của tổ chức là riêng tư. Giá trị này bị bỏ qua nếu repo đã tồn tại.
- token (str , optional) --
  Mã thông báo xác thực tùy chọn cho Hugging Face Hub. Nếu không có mã thông báo nào được chuyển, mặc định
  vào mã thông báo được lưu cục bộ khi đăng nhập bằng deathface-cli login . Sẽ gây ra lỗi
  nếu không có mã thông báo nào được chuyển và người dùng chưa đăng nhập.
- revision (str , optional) --
  Branch để đẩy các tập tin đã tải lên. Mặc định là nhánh "chính".
- create_pr (bool , optional, defaults to False) --
  Có nên tạo PR bằng các tệp đã tải lên hay cam kết trực tiếp.
- num_shards (Dict[str, int] , optional) --
  Số lượng mảnh để viết. Theo mặc định, bằng .num_shards của tập dữ liệu này.
  Sử dụng từ điển để xác định num_shards khác nhau cho mỗi phần tách.
- embed_external_files (bool , defaults to True) --

Có nhúng byte tệp vào phân đoạn hay không.
Cụ thể, điều này sẽ thực hiện những việc sau trước khi đẩy các trường thuộc loại:
Âm thanh và Hình ảnh xóa thông tin đường dẫn cục bộ và nhúng nội dung tệp vào
Tập tin sàn gỗ.

- num_proc (int , optional, defaults to None) --
  Số lượng quy trình khi chuẩn bị và tải lên tập dữ liệu.
  Điều này hữu ích nếu tập dữ liệu được tạo từ nhiều mẫu hoặc tệp phương tiện để nhúng.
  Đa xử lý bị tắt theo mặc định.
  0huggingface_hub.CommitInfo
  Đẩy DatasetDict vào trung tâm dưới dạng tập dữ liệu Parquet.
  DatasetDict được đẩy bằng các yêu cầu HTTP và không cần phải có git hoặc
  git-lfs đã được cài đặt.

Mỗi phần chia tập dữ liệu sẽ được đẩy độc lập. Tập dữ liệu được đẩy sẽ giữ nguyên phần phân chia ban đầu những cái tên.

Theo mặc định, các tệp Parquet kết quả được tự chứa: nếu tập dữ liệu của bạn chứa Hình ảnh hoặc Âm thanh
data, tệp Parquet sẽ lưu trữ byte hình ảnh hoặc tệp âm thanh của bạn.
Bạn có thể tắt tính năng này bằng cách đặt embed_external_files thành Sai.

Ví dụ:

```
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>")
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>", private=True)
>>> dataset_dict.push_to_hub("<organization>/<dataset_id>", num_shards={"train": 1024, "test": 8})
```

If you want to add a new configuration (or subset) to a dataset (e.g. if the dataset has multiple tasks/versions/languages):

```
>>> english_dataset.push_to_hub("<organization>/<dataset_id>", "en")
>>> french_dataset.push_to_hub("<organization>/<dataset_id>", "fr")
>>> # later
>>> english_dataset = load_dataset("<organization>/<dataset_id>", "en")
>>> french_dataset = load_dataset("<organization>/<dataset_id>", "fr")
```

Tính năngdatasets.Features

bộ dữ liệu lớp.Featuresdatasets.Featureshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L1734[{"name": "*args", "val": ""}, {"name": "**kwargs", "val": ""}]
Một từ điển đặc biệt xác định cấu trúc bên trong của tập dữ liệu.

Instantiated with a dictionary of type dict[str, FieldType] , where keys are the desired
tên cột,
và giá trị là loại của cột đó.

FieldType có thể là một trong những loại sau:

- Tính năng giá trị chỉ định một giá trị kiểu dữ liệu duy nhất, ví dụ: int64 hoặc chuỗi.
- Tính năng ClassLabel chỉ định một tập hợp các lớp được xác định trước có thể có các nhãn được liên kết
  cho họ và
  sẽ được lưu trữ dưới dạng số nguyên trong tập dữ liệu.
- Lệnh Python chỉ định một tính năng tổng hợp chứa ánh xạ các trường con tới trường con
  đặc trưng.
  Có thể có các trường lồng nhau của các trường lồng nhau một cách tùy ý.
- Danh sách hoặc Danh sách lớn chỉ định một tính năng tổng hợp chứa một chuỗi
  các tính năng phụ, tất cả đều có cùng loại tính năng.
- Tính năng Array2D, Array3D, Array4D hoặc Array5D cho mảng đa chiều.
- Tính năng âm thanh lưu trữ đường dẫn tuyệt đối tới một tập tin âm thanh hoặc từ điển với đường dẫn tu
  con đường
  to an audio file ("path" key) and its bytes content ("bytes" key).
  Tính năng này tải âm thanh một cách lười biếng bằng bộ giải mã.
- Tính năng hình ảnh để lưu trữ đường dẫn tuyệt đối tới một tập tin hình ảnh, một đối tượng np.ndarray, r
  Đối tượng PIL.Image.Image
  or a dictionary with the relative path to an image file ("path" key) and its bytes content
  ("bytes" key).
  Tính năng này trích xuất dữ liệu hình ảnh.
- Tính năng video để lưu trữ đường dẫn tuyệt đối tới một tập tin video, một
  Đối tượng torchcodec.decodings.VideoDecoding
  or a dictionary with the relative path to a video file ("path" key) and its bytes content
  ("bytes" key).

Tính năng này tải video một cách lười biếng bằng bộ giải mã.

- Tính năng Pdf lưu trữ đường dẫn tuyệt đối tới tệp PDF, đối tượng pdfplumber.pdf.PDF or a dictionary with the relative path to a PDF file ("path" key) and its bytes content ("bytes" key).
  Tính năng này tải tệp PDF một cách lười biếng bằng trình đọc PDF.
- Tính năng Dịch thuật hoặc DịchVariableLanguages dành riêng cho Dịch máy.

copydatasets.Features.copyhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2157[]Features

Tạo một bản sao sâu của Tính năng.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> copy_of_features = ds.features.copy()
>>> copy_of_features
{'label': ClassLabel(names=['neg', 'pos']),
 'text': Value('string')}
```

giải mã_batchdatasets.Features.decode_batchhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2130[{"name": "batch", "val": ": dict"}, {"name": "token_per_repo_id", "val": ": typing.Optional[dict[str, typing.Union[str, bool, NoneType]]] = None"}]- batch (dict[str, list[Any]]) --
Dữ liệu hàng loạt tập dữ liệu.

- token_per_repo_id (dict , optional) --
  Để truy cập và giải mã các tệp âm thanh hoặc hình ảnh từ kho lưu trữ riêng trên Hub, bạn có thể vượt qua
  a dictionary repo_id (str) -> token (bool or str)0 dict[str, list[Any]]
  Giải mã hàng loạt với tính năng giải mã tùy chỉnh.

giải mã_columndatasets.Features.decode_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2105[{"name": "column", "val": ": list"}, {"name": "column_name", "val": ": str"}, {"name": "token_per_repo_id", "val": ": typing.Optional[dict[str, typing.Union[str, bool, NoneType]]] = None"}]- column (list[Any]) --

Dữ liệu cột tập dữ liệu.

- column_name (str) --
  Dataset column name.0 list[Any]
  Cột giải mã với tính năng giải mã tùy chỉnh.

giải mã_exampledatasets.Features.decode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2082[{"name": "example", "val": ": dict"}, {"name": "token_per_repo_id", "val": ": typing.Optional[dict[str, typing.Union[str, bool, NoneType]]] = None"}]- example (dict[str, Any]) --
Dữ liệu hàng tập dữ liệu.

- token_per_repo_id (dict , optional) --
  Để truy cập và giải mã các tệp âm thanh hoặc hình ảnh từ kho lưu trữ riêng trên Hub, bạn có thể vượt qua
  a dictionary repo_id (str) -> token (bool or str) .0 dict[str, Any]
  Ví dụ giải mã với giải mã tính năng tùy chỉnh.

Encode_batchdatasets.Features.encode_batchhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2063[{"name": "batch", "val": ""}]- batch (dict[str, list[Any]]) --
Data in a Dataset batch.0 dict[str, list[Any]]

Mã hóa hàng loạt thành định dạng cho Arrow.

Encode_columndatasets.Features.encode_columnhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2047[{"name": "column", "val": ""}, {"name": "column_name", "val": ": str"}]- column (list[Any]) --
Dữ liệu trong cột Tập dữ liệu.

- column_name (str) --
  Dataset column name.0 list[Any]

Mã hóa cột thành định dạng cho Mũi tên.

Encode_exampledatasets.Features.encode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2033[{"name": "example", "val": ""}]- example (dict[str, Any]) --

Data in a Dataset row.0 dict[str, Any]

Mã hóa ví dụ thành định dạng cho Arrow.

Flattendatasets.Features.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L2230[{"name": "max_depth", "val": " = 16"}]FeaturesThe đặc điểm phẳng.
Làm phẳng các tính năng. Mỗi cột từ điển sẽ bị xóa và được thay thế bằng
tất cả các trường con chứa nó. Các trường mới được đặt tên bằng cách nối các trường
name of the original column and the subfield name like this: <original>.<subfield> .

Nếu một cột chứa các từ điển lồng nhau thì tất cả các tên trường con cấp thấp hơn sẽ được also concatenated to form new columns: <original>.<subfield>.<subsubfield> , etc.

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("rajpurkar/squad", split="train")
>>> ds.features.flatten()
{'answers.answer_start': List(Value('int32'), id=None),
 'answers.text': List(Value('string'), id=None),
 'context': Value('string'),
 'id': Value('string'),
 'question': Value('string'),
 'title': Value('string')}
```

from_arrow_schemadatasets.Features.from_arrow_schemahttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L1816[{"name": "pa_schema", "val": ": Schema"}]- pa_schema (pyarrow.Schema) --
Lược đồ mũi tên.0Tính năng

Xây dựng các tính năng từ lược đồ mũi tên.
Nó cũng kiểm tra siêu dữ liệu lược đồ để tìm các tính năng của Bộ dữ liệu Ôm khuôn mặt.
Các trường không thể rỗng không được hỗ trợ và được đặt thành có thể rỗng.

Ngoài ra, pa.dictionary không được hỗ trợ và thay vào đó nó sử dụng loại cơ bản của nó.
Do đó, các bộ dữ liệu chuyển đổi các đối tượng từ điển thành giá trị thực của chúng.

from_dictdatasets.Features.from_dicthttps://github.com/huggingface/datasets/blob/4.2.0/src/

datasets/features/features.py#L1850[{"name": "dic", "val": ""}]- dic (dict[str, Any]) --
Từ điển Python.0Tính năng

Construct [Features] from dict.

Tạo lại đối tượng tính năng lồng nhau từ một lệnh đã được giải tuần tự hóa.
Chúng tôi sử dụng khóa _type để suy ra tên lớp dữ liệu của FieldType đối tượng.

Nó cho phép một cú pháp xây dựng thuận tiện
để xác định các tính năng từ các từ điển JSON đã được giải tuần tự hóa. Chức năng này được sử dụng đặc biệ
khử lưu huỳnh
a [DatasetInfo] that was dumped to a JSON object. This acts as an analogue to
[Features.from_arrow_schema] and handles the recursive field-by-field instantiation, but
không yêu cầu
bất kỳ ánh xạ nào đến/từ pyarrow, ngoại trừ thực tế là nó tận dụng ánh xạ của
pyarrow nguyên thủy
dtypes that [Value] automatically performs.

Ví dụ:

```
>>> Features.from_dict(&amp;lcub;'_type': &amp;lcub;'dtype': 'string', 'id': None, '_type': 'Value
&amp;lcub;'_type': Value('string')}
```

reorder_fields_asdatasets.Features.reorder_fields_ashttps://github.com/huggingface/datasets/
blob/4.2.0/src/datasets/features/features.py#L2177[{"name": "other", "val": ": Features"}]- other
([Features]) --
The other [Features] to align with.0[Features]

Reorder Features fields to match the field order of other [Features].

Thứ tự của các trường rất quan trọng vì nó quan trọng đối với dữ liệu mũi tên cơ bản.
Việc sắp xếp lại các trường cho phép làm cho kiểu dữ liệu mũi tên bên dưới khớp với nhau.

Ví dụ:

```
>>> from datasets import Features, List, Value
>>> # let's say we have two features with a different order of nested fields (for a and b for exam
>>> f1 = Features(&amp;lcub;"root": &amp;lcub;"a": Value("string"), "b": Value("string")}})
>>> f2 = Features(&amp;lcub;"root": &amp;lcub;"b": Value("string"), "a": Value("string")}})
>>> assert f1.type != f2.type
>>> # re-ordering keeps the base structure (here List is defined at the root level), but makes the
>>> f1.reorder_fields_as(f2)
&amp;lcub;'root': List(&amp;lcub;'b': Value('string'), 'a': Value('string')})}
>>> assert f1.reorder_fields_as(f2).type == f2.type
```

Scalardatasets.Value

bộ dữ liệu lớp.Valuedatasets.Valuehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L482[{"name": "dtype", "val": ": str"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- dtype (str) --
Tên kiểu dữ liệu.0

Giá trị đặc trưng vô hướng của một kiểu dữ liệu cụ thể.

Các loại giá trị có thể có như sau:

- vô giá trị
- bool
- bạn8
- int16
- int32
- int64
- uint8
- uint16
- uint32
- uint64
- phao16
- float32 (alias float)
- float64 (alias double)
- time32[(s|ms)]
- time64[(us|ns)]

- timestamp[(s|ms|us|ns)]
- timestamp[(s|ms|us|ns), tz=(tzstring)]
- ngày32
- ngày64
- duration[(s|ms|us|ns)]
- decimal128(precision, scale)
- decimal256(precision, scale)
- nhị phân
- lớn_nhị phân
- sợi dây
- chuỗi_lớn
- chuỗi_view

Ví dụ:

```
>>> from datasets import Features
>>> features = Features({'stars': Value('int32')})
>>> features
{'stars': Value('int32')}
```

bộ dữ liệu lớp.ClassLabeldatasets.ClassLabelhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L973[{"name": "num_classes", "val": ": dataclasses.InitVar[typing.Optional[int]] = None"}, {"name": "names", "val": ": list = None"}, {"name": "names_file", "val": ": dataclasses.InitVar[typing.Optional[str]] = None"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- num_classes (int , optional) --
Number of classes. All labels must be < num_classes .

- names (list of str , optional) --
  Tên chuỗi cho các lớp số nguyên.
  Thứ tự cung cấp tên được giữ nguyên.
- names_file (str , optional) --
  Đường dẫn đến tệp có tên cho các lớp số nguyên, mỗi tên một dòng.0
  Loại tính năng cho nhãn lớp số nguyên.

Có 3 cách để xác định ClassLabel , tương ứng với 3 đối số:

- num_classes : Create 0 to (num_classes-1) labels.

• tên: Danh sách các chuỗi nhãn.
•name_file : File chứa danh sách các nhãn.

Dưới mui xe, các nhãn được lưu trữ dưới dạng số nguyên.
Bạn có thể sử dụng số nguyên âm để biểu thị các nhãn không xác định/thiếu.

Ví dụ:

```
>>> from datasets import Features, ClassLabel
>>> features = Features({'label': ClassLabel(num_classes=3, names=['bad', 'ok', 'good'])})
>>> features
{'label': ClassLabel(names=['bad', 'ok', 'good'])}
```

cast_storagedatasets.ClassLabel.cast_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L1138[{"name": "storage", "val": ": typing.Union[pyarrow.lib.StringArray, pyarrow.lib.IntegerArray]"}]- storage (Union[pa.StringArray, pa.IntegerArray]) --
Mảng PyArrow để truyền.0 pa.Int64Array Mảng trong kiểu lưu trữ mũi tên ClassLabel.
Truyền một mảng Mũi tên tới loại lưu trữ mũi tên ClassLabel.
Các loại Mũi tên có thể được chuyển đổi thành loại lưu trữ pyarrow ClassLabel là:

- pa.string()
- pa.int()

int2strdatasets.ClassLabel.int2strhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L1092[{"name": "values", "val": ": typing.Union[int, collections.abc.Iterable]"}]
Conversion integer => class name string .

Về nhãn không xác định/thiếu: truyền số nguyên âm sẽ tăng ValueError .

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> ds.features["label"].int2str(0)
'phủ định'
```

str2intdatasets.ClassLabel.str2inthttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L1047[{"name": "values", "val": ": typing.Union[str, collections.abc.Iterable]"}]
Conversion class name string => integer .

Ví dụ:

```
>>> from datasets import load_dataset
>>> ds = load_dataset("cornell-movie-review-data/rotten_tomatoes", split="train")
>>> ds.features["label"].str2int('neg')
0
```

Compositedatasets.LargeList

bộ dữ liệu lớp.LargeListdatasets.LargeListhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L1232[{"name": "feature", "val": ": typing.Any"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- feature (FeatureType) --
Kiểu dữ liệu tính năng con của từng mục trong danh sách lớn.0
Loại tính năng cho dữ liệu danh sách lớn bao gồm loại dữ liệu tính năng con.

Nó được hỗ trợ bởi pyarrow.LargeListType , giống như pyarrow.ListType nhưng với 64-bit hơn hơn độ lệch 32-bit.

bộ dữ liệu lớp.Listdatasets.Listhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L1204[{"name": "feature", "val": ": typing.Any"}, {"name": "length", "val": ": int = -1"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- feature (FeatureType) --
Kiểu dữ liệu tính năng con của từng mục trong danh sách lớn.

- length (optional int , default to -1) --
  Độ dài của danh sách nếu nó được cố định.
  Mặc định là -1 có nghĩa là độ dài tùy ý.0
  Loại tính năng cho dữ liệu danh sách lớn bao gồm loại dữ liệu tính năng con.

Nó được hỗ trợ bởi pyarrow.ListType, sử dụng độ lệch 32 bit hoặc độ dài cố định.

bộ dữ liệu lớp.Sequencedatasets.Sequencehttps://github.com/huggingface/datasets/blob/

4.2.0/src/datasets/features/features.py#L1170[{"name": "feature", "val": " = None"}, {"name": "length", "val": " = -1"}, {"name": "**kwargs", "val": ""}]- feature (FeatureType) --
Kiểu dữ liệu tính năng con của từng mục trong danh sách lớn.

- length (optional int , default to -1) --
  Độ dài của danh sách nếu nó được cố định.
  Mặc định là -1 có nghĩa là độ dài tùy ý.0Danh sách tính năng được chỉ định, ngoại trừ dict
  các tính năng phụ
  được chuyển đổi thành mệnh lệnh danh sách các tính năng phụ để tương thích với TFDS.

Sequence là một tiện ích tự động chuyển đổi tính năng từ điển nội bộ thành từ điển của
danh sách. Hành vi này được triển khai để có lớp tương thích với Bộ dữ liệu TensorFlow
thư viện nhưng có thể
không mong muốn trong một số trường hợp. Nếu bạn không muốn hành vi này, bạn có thể sử dụng Danh s
thay vì Trình tự.

Translationdatasets.Translation

tập dữ liệu lớp.Translationdatasets.Translationhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/translation.py#L12[{"name": "languages", "val": ": list"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- languages (dict) --
Một từ điển cho mỗi ví dụ ánh xạ mã ngôn ngữ chuỗi sang bản dịch chuỗi.0
Tính năng dịch với các ngôn ngữ cố định cho mỗi ví dụ.
Ở đây để tương thích với tfds.

Ví dụ:

```
>>> # At construction time:
>>> datasets.features.Translation(languages=['en', 'fr', 'de'])
>>> # During data generation:
>>> yield {
...'en': 'con mèo',
...'en': 'con mèo',
...'en': 'con mèo'
... }
```

Flattendatasets.Translation.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/

datasets/features/translation.py#L44[]
Làm phẳng tính năng Dịch thành từ điển.

lớp học
bộ dữ liệu.TranslationVariableLanguagesdatasets.TranslationVariableLanguageshttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/translation.py#L52[{"name": "languages", "val": ": typing.Optional[list] = None"}, {"name": "num_languages", "val": ": typing.Optional[int] = None"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- languages (dict) --
Một từ điển cho mỗi ví dụ ánh xạ mã ngôn ngữ chuỗi thành một hoặc nhiều chuỗi các bản dịch.
Các ngôn ngữ hiện tại có thể khác nhau tùy theo từng ví dụ.0- ngôn ngữ hoặc bản dịch (variable-length 1D tf.Tensor of tf.string)Language codes sorted in ascending order or bản dịch văn bản đơn giản, được sắp xếp để phù hợp với mã ngôn ngữ.
Tính năng dịch với nhiều ngôn ngữ khác nhau cho mỗi ví dụ.
Ở đây để tương thích với tfds.

Ví dụ:

```
>>> # At construction time:
>>> datasets.features.TranslationVariableLanguages(languages=['en', 'fr', 'de'])
>>> # During data generation:
>>> yield {
...'en': 'con mèo',
...'fr': ['le chat', 'la chatte,']
...'en': 'con mèo'
... }
>>> # Tensor returned :
>>> {
...'language': ['en', 'de', 'fr', 'fr'],
...'translation': ['the cat', 'die katze', 'la chatte', 'le chat'],
... }
```

Flattendatasets.TranslationVariableLanguages.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/translation.py#L122[]
Làm phẳng tính năng TranslationVariableLanguages thành một từ điển.

Mảng dữ liệu.Array2D

bộ dữ liệu lớp.Array2Ddatasets.Array2Dhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L578[{"name": "shape", "val": ": tuple"}, {"name": "dtype", "val": ": str"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- shape (tuple) --
Kích thước của mỗi chiều.

- dtype (str) --
  Tên kiểu dữ liệu.0
  Tạo mảng hai chiều.

Ví dụ:

```
>>> from datasets import Features
>>> features = Features({'x': Array2D(shape=(1, 3), dtype='int32')})
```

bộ dữ liệu lớp.Array3Ddatasets.Array3Dhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L603[{"name": "shape", "val": ": tuple"}, {"name": "dtype", "val": ": str"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- shape (tuple) --
Kích thước của mỗi chiều.

- dtype (str) --
  Tên kiểu dữ liệu.0
  Tạo một mảng ba chiều.

Ví dụ:

```
>>> from datasets import Features
>>> features = Features({'x': Array3D(shape=(1, 2, 3), dtype='int32')})
```

bộ dữ liệu lớp.Array4Ddatasets.Array4Dhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L628[{"name": "shape", "val": ": tuple"}, {"name": "dtype", "val": ": str"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- shape (tuple) --
Kích thước của mỗi chiều.

- dtype (str) --
  Tên kiểu dữ liệu.0

Tạo một mảng bốn chiều.

Ví dụ:

```
>>> from datasets import Features
>>> features = Features({'x': Array4D(shape=(1, 2, 2, 3), dtype='int32')})
```

bộ dữ liệu lớp.Array5Ddatasets.Array5Dhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/features.py#L653[{"name": "shape", "val": ": tuple"}, {"name": "dtype", "val": ": str"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- shape (tuple) --
Kích thước của mỗi chiều.

- dtype (str) --
  Tên kiểu dữ liệu.0
  Tạo một mảng năm chiều.

Ví dụ:

```
>>> from datasets import Features
>>> features = Features({'x': Array5D(shape=(1, 2, 2, 3, 3), dtype='int32')})
```

Audiodatasets.Audio

bộ dữ liệu lớp.Audiodatasets.Audiohttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/audio.py#L24[{"name": "sampling_rate", "val": ": typing.Optional[int] = None"}, {"name": "decode", "val": ": bool = True"}, {"name": "stream_index", "val": ": typing.Optional[int] = None"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- sampling_rate (int , optional) --
Tỷ lệ lấy mẫu mục tiêu Nếu Không , tốc độ lấy mẫu gốc sẽ được sử dụng.

- mono (bool , defaults to True) --
  Có chuyển đổi tín hiệu âm thanh thành đơn âm hay không bằng cách lấy trung bình các mẫu trên các kênh.
- decode (bool , defaults to True) --
  Có giải mã dữ liệu âm thanh hay không. Nếu sai ,
  trả về từ điển cơ bản ở định dạng

{"path": audio_path, "bytes": audio_bytes} .

- stream_index (int , optional) --

  Chỉ mục phát trực tuyến để sử dụng từ tệp. Nếu Không có giá trị mặc định là chỉ số "tốt nhất".0

  Tính năng âm thanh để trích xuất dữ liệu âm thanh từ một tệp âm thanh.

Đầu vào: Tính năng Âm thanh chấp nhận làm đầu vào:

- A str : Absolute path to the audio file (i.e. random access is allowed).
- A pathlib.Path : path to the audio file (i.e. random access is allowed).
- Một lệnh với các phím:

  đường dẫn : Chuỗi có đường dẫn tương đối của tệp âm thanh đến tệp lưu trữ.

  byte : Byte nội dung của tệp âm thanh.

  Điều này hữu ích cho các tệp parquet hoặc webdataset nhúng tệp âm thanh.
- Một lệnh với các phím:

  array : Mảng chứa mẫu âm thanh

  samples_rate : Số nguyên tương ứng với tốc độ lấy mẫu của mẫu âm thanh.
- A torchcodec.decodings.AudioDecoding : đối tượng giải mã âm thanh torchcodec.

Đầu ra: Âm thanh có dữ liệu đầu ra dưới dạng các đối tượng torchcodec.decodings.AudioDecoding, với phím bổ sung:

- array : Mảng chứa mẫu âm thanh
- samples_rate : Số nguyên tương ứng với tốc độ lấy mẫu của mẫu âm thanh.

Ví dụ:

```
>>> from datasets import load_dataset, Audio
>>> ds = load_dataset("PolyAI/minds14", name="en-US", split="train")
>>> ds = ds.cast_column("audio", Audio(sampling_rate=44100))
>>> ds[0]["audio"]
<datasets.features._torchcodec.AudioDecoder object at 0x11642b6a0>
>>> audio = ds[0]["audio"]
>>> audio.get_samples_played_in_range(0, 10)
Mẫu âm thanh:
    data (shape): torch.Size([2, 110592])
    pts_giây: 0,0
    thời lượng_giây: 2,507755102040816
    sample_rate: 44100
```

cast_storagedatasets.Audio.cast_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/audio.py#L223[{"name": "storage", "val": ": typing.Union[pyarrow.lib.StringArray, pyarrow.lib.StructArray]"}]- storage (Union[pa.StringArray, pa.StructArray]) --
Mảng PyArrow thành cast.0 pa.StructArray Mảng trong kiểu lưu trữ mũi tên Âm thanh, nghĩa là pa.struct({"bytes": pa.binary(), "path": pa.string()})
Truyền mảng Mũi tên sang loại lưu trữ mũi tên Âm thanh.
Các loại Mũi tên có thể được chuyển đổi thành loại lưu trữ Audio pyarrow là:

- pa.string() - it must contain the "path" data
- pa.binary() - it must contain the audio bytes
- pa.struct({"bytes": pa.binary()})
- pa.struct({"path": pa.string()})
- pa.struct({"bytes": pa.binary(), "path": pa.string()}) - order doesn't matter

giải mã_exampledatasets.Audio.decode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/audio.py#L159[{"name": "value", "val": ": dict"}, {"name": "token_per_repo_id", "val": ": typing.Optional[dict[str, typing.Union[str, bool, NoneType]]] = None"}]- value (dict) --
Một từ điển có khóa:

- path : Chuỗi có đường dẫn file âm thanh tương đối.
- byte: Byte của tập tin âm thanh.
- token_per_repo_id (dict , optional) --
  Để truy cập và giải mã
  tệp âm thanh từ kho riêng trên Hub, bạn có thể chuyển
  a dictionary repo_id (str) -> token (bool or str)0 torchcodec.decoders.AudioDecoder
  Giải mã tập tin âm thanh ví dụ thành dữ liệu âm thanh.

embed_storagedatasets.Audio.embed_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/audio.py#L263[{"name": "storage", "val": ": StructArray"}, {"name": "token_per_repo_id", "val": " = None"}]- storage (pa.StructArray) --
Mảng PyArrow thành embed.0 pa.StructArray Mảng trong kiểu lưu trữ mũi tên Âm thanh, nghĩa là pa.struct({"bytes": pa.binary(), "path": pa.string()}) .
Nhúng tệp âm thanh vào mảng Mũi tên.

Encode_exampledatasets.Audio.encode_examplehttps://github.com/huggingface/datasets/

blob/4.2.0/src/datasets/features/audio.py#L91[{"name": "value", "val": ": typing.Union[str, bytes, bytearray, dict, ForwardRef('AudioDecoder')]"}]- value (str , bytes , bytearray , dict , AudioDecoder) --
Dữ liệu được truyền dưới dạng đầu vào cho tính năng Âm thanh.0 dict
Mã hóa ví dụ thành định dạng cho Arrow.

Flattendatasets.Audio.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/audio.py#L212[]
Nếu ở trạng thái có thể giải mã được, hãy đưa ra lỗi, nếu không thì sẽ đưa đối tượng địa lý vào từ điển.

Bộ dữ liệu hình ảnh.Hình ảnh

bộ dữ liệu lớp.Imagedatasets.Imagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/image.py#L47[{"name": "mode", "val": ": typing.Optional[str] = None"}, {"name": "decode", "val": ": bool = True"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- mode (str , optional) --
Chế độ để chuyển đổi hình ảnh sang. Nếu Không có, chế độ gốc của hình ảnh sẽ được sử dụng.

- decode (bool , defaults to True) --
  Có giải mã dữ liệu hình ảnh hay không. Nếu sai ,
  trả về từ điển cơ bản ở định dạng
  {"path": image_path, "bytes": image_bytes} .0
  Tính năng hình ảnh để đọc dữ liệu hình ảnh từ một tập tin hình ảnh.

Đầu vào: Tính năng Hình ảnh chấp nhận làm đầu vào:

- A str : Absolute path to the image file (i.e. random access is allowed).
- A pathlib.Path : path to the image file (i.e. random access is allowed).
- Một lệnh với các phím:
    path : Chuỗi chứa đường dẫn tương đối của file ảnh tới file lưu trữ.
    byte : Byte của tệp hình ảnh.
  Điều này hữu ích cho các tệp parquet hoặc webdataset nhúng tệp hình ảnh.
- Np.darray: hình ảnh chạy đua.
- A PIL.Image.Image : Đối tượng hình ảnh PIL.

Đầu ra: Hình ảnh có dữ liệu đầu ra dưới dạng đối tượng PIL.Image.Image.

Ví dụ:

```
>>> from datasets import load_dataset, Image
>>> ds = load_dataset("AI-Lab-Makerere/beans", split="train")
>>> ds.features["image"]
Image(decode=True, id=None)
>>> ds[0]["image"]
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x500 at 0x15E52E7F0>
>>> ds = ds.cast_column('image', Image(decode=False))
{'bytes': None,
 'đường dẫn': '/root/.cache/huggingface/datasets/downloads/extracted/b0a21163f78769a2cf11f58dfc767f
```

cast_storagedatasets.Image.cast_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/image.py#L213[{"name": "storage", "val": ": typing.Union[pyarrow.lib.StringArray, pyarrow.lib.StructArray, pyarrow.lib.ListArray]"}]- storage (Union[pa.StringArray, pa.StructArray, pa.ListArray]) --
Mảng PyArrow thành cast.0 pa.StructArray Mảng trong kiểu lưu trữ Mũi tên hình ảnh, nghĩa là pa.struct({"bytes": pa.binary(), "path": pa.string()}) .
Truyền một mảng Mũi tên sang loại lưu trữ Mũi tên hình ảnh.
Các loại Mũi tên có thể được chuyển đổi thành loại lưu trữ Hình ảnh pyarrow là:

- pa.string() - it must contain the "path" data
- pa.binary() - it must contain the image bytes
- pa.struct({"bytes": pa.binary()})
- pa.struct({"path": pa.string()})
- pa.struct({"bytes": pa.binary(), "path": pa.string()}) - order doesn't matter
- pa.list(*) - it must contain the image array data

giải mã_exampledatasets.Image.decode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/image.py#L139[{"name": "value", "val": ": dict"}, {"name": "token_per_repo_id", "val": " = None"}]- value (str or dict) --
Một chuỗi có đường dẫn tệp hình ảnh tuyệt đối, một từ điển có
phím:

- path : Chuỗi có đường dẫn file ảnh tuyệt đối hoặc tương đối.
- byte: Các byte của file ảnh.
- token_per_repo_id (dict , optional) --

Để truy cập và giải mã
tập tin hình ảnh từ kho riêng trên Hub, bạn có thể chuyển
a dictionary repo_id (str) -> token (bool or str).0 PIL.Image.Image
Giải mã tập tin hình ảnh ví dụ thành dữ liệu hình ảnh.

embed_storagedatasets.Image.embed_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/image.py#L259[{"name": "storage", "val": ": StructArray"}, {"name": "token_per_repo_id", "val": " = None"}]- storage (pa.StructArray) --
Mảng PyArrow để embed.0 pa.StructArray Mảng trong kiểu lưu trữ Mũi tên hình ảnh, nghĩa là pa.struct({"bytes": pa.binary(), "path": pa.string()}) .
Nhúng tập tin hình ảnh vào mảng Mũi tên.

Encode_exampledatasets.Image.encode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/image.py#L98[{"name": "value", "val": ": typing.Union[str, bytes, bytearray, dict, numpy.ndarray, ForwardRef('PIL.Image.Image')]"}]- value (str , np.ndarray , PIL.Image.Image or dict) --
Dữ liệu được truyền dưới dạng đầu vào cho tính năng Hình ảnh.0 dict với các trường "đường dẫn" và "byte"
Mã hóa ví dụ thành định dạng cho Arrow.

Flattendatasets.Image.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/image.py#L200[]
Nếu ở trạng thái có thể giải mã được, hãy trả về chính đối tượng địa lý đó, nếu không thì làm phẳng đối tượ

Bộ dữ liệu video.Video

bộ dữ liệu lớp.Videodatasets.Videohttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/video.py#L29[{"name": "decode", "val": ": bool = True"}, {"name": "stream_index", "val": ": typing.Optional[int] = None"}, {"name": "dimension_order", "val": ": typing.Literal['NCHW', 'NHWC'] = 'NCHW'"}, {"name": "num_ffmpeg_threads", "val": ": int = 1"}, {"name": "device", "val": ": typing.Union[str, ForwardRef('torch.device'), NoneType] = 'cpu'"}, {"name": "seek_mode", "val": ": typing.Literal['exact', 'approximate'] = 'exact'"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- mode (str , optional) --
Chế độ chuyển đổi video sang. Nếu Không có thì chế độ gốc của video sẽ được sử dụng.

- decode (bool , defaults to True) --
  Có giải mã dữ liệu video hay không. Nếu sai ,
  trả về từ điển cơ bản ở định dạng

{"path": video_path, "bytes": video_bytes} .

- stream_index (int , optional) --
  Chỉ mục phát trực tuyến để sử dụng từ tệp. Nếu Không có mặc định là chỉ mục "tốt nhất".
- dimension_order (str , defaults to NCHW) --
  Thứ tự kích thước của các khung được giải mã.
  trong đó N là kích thước lô, C là số lượng kênh,
  H là chiều cao và W là chiều rộng của khung.
- num_ffmpeg_threads (int , defaults to 1) --
  The number of threads to use for decoding the video. (Recommended to keep this at 1)
- device (str or torch.device , defaults to cpu) --
  Thiết bị được sử dụng để giải mã video.
- seek_mode (str , defaults to exact) --
  Xác định xem quyền truy cập khung sẽ là “chính xác” hay “gần đúng”.
  Đảm bảo chính xác rằng việc yêu cầu khung i sẽ luôn trả về khung i, nhưng làm như vậy đòi hỏi
  lần quét đầu tiên của tập tin.
  Gần đúng nhanh hơn vì nó tránh được việc quét tệp, nhưng kém chính xác hơn vì nó sử dụng thuộc tính
  siêu dữ liệu để tính toán xem tôi có thể ở đâu.
  đọc thêm tại đây0

Tính năng Video để đọc dữ liệu video từ một tập tin video.

Đầu vào: Tính năng Video chấp nhận làm đầu vào:

- A str : Absolute path to the video file (i.e. random access is allowed).
- A pathlib.Path : path to the video file (i.e. random access is allowed).
- Một lệnh với các phím:
  đường dẫn : Chuỗi có đường dẫn tương đối của tệp video trong kho lưu trữ dữ liệu.
  byte : Byte của tệp video.
  Điều này hữu ích cho các tệp parquet hoặc webdataset nhúng tệp video.
- A torchcodec.decodings.VideoDecoding : đối tượng giải mã video torchcodec.

Đầu ra: Video có dữ liệu đầu ra dưới dạng đối tượng torchcodec.decodings.VideoDecoding.

Ví dụ:

```
>>> from datasets import Dataset, Video
>>> ds = Dataset.from_dict({"video":["path/to/Screen Recording.mov"]}).cast_column("video", Video(
>>> ds.features["video"]
Video(decode=True, id=None)
>>> ds[0]["video"]
<torchcodec.decoders._video_decoder.VideoDecoder object at 0x14a61e080>
>>> video = ds[0]["video"]
>>> video.get_frames_in_range(0, 10)
FrameBatch:
data (shape): torch.Size([10, 3, 50, 66])
pts_seconds: tensor([0.4333, 0.4333, 0.4333, 0.4333, 0.4333, 0.4333, 0.4333, 0.4333, 0.4333,
        0.4333], dtype=torch.float64)
duration_seconds: tensor([0.0167, 0.0167, 0.0167, 0.0167, 0.0167, 0.0167, 0.0167, 0.0167, 0.0167,
        0.0167], dtype=torch.float64)
>>> ds.cast_column('video', Video(decode=False))[0]["video]
{'bytes': None,
 'path': 'path/to/Screen Recording.mov'}
```

cast_storagedatasets.Video.cast_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/video.py#L241[{"name": "storage", "val": ": typing.Union[pyarrow.lib.StringArray, pyarrow.lib.StructArray, pyarrow.lib.ListArray]"}]- storage (Union[pa.StringArray, pa.StructArray, pa.ListArray]) --
Mảng PyArrow thành cast.0 pa.StructArray Mảng trong kiểu lưu trữ mũi tên Video, tức là pa.struct({"bytes": pa.binary(), "path": pa.string()}) .
Truyền mảng Mũi tên sang loại lưu trữ mũi tên Video.
Các loại Mũi tên có thể được chuyển đổi thành loại lưu trữ Video pyarrow là:

- pa.string() - it must contain the "path" data
- pa.binary() - it must contain the video bytes
- pa.struct({"bytes": pa.binary()})
- pa.struct({"path": pa.string()})
- pa.struct({"bytes": pa.binary(), "path": pa.string()}) - order doesn't matter
- pa.list(*) - it must contain the video array data

giải mã_exampledatasets.Video.decode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/video.py#L155[{"name": "value", "val": ": typing.Union[str, datasets.features.video.Example]"}, {"name": "token_per_repo_id", "val": ":

typing.Optional[dict[str, typing.Union[bool, str]]] = None"}]- value (str or dict) --
Một chuỗi có đường dẫn tệp video tuyệt đối, một từ điển có
phím:

- path : Chuỗi có đường dẫn file video tuyệt đối hoặc tương đối.
- byte : Các byte của file video.
- token_per_repo_id (dict , optional) --
  Để truy cập và giải mã
  tệp video từ kho riêng trên Hub, bạn có thể chuyển
  a dictionary repo_id (str) -> token (bool or str).0 torchcodec.decoders.VideoDecoder
  Giải mã tập tin video ví dụ thành dữ liệu video.

Encode_exampledatasets.Video.encode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/video.py#L107[{"name": "value", "val": ": typing.Union[str, byte, bytearray, tập dữ liệu.features.video.Example, numpy.ndarray, ForwardRef('VideoDecoder')]"}]- value (str , np.ndarray , bytes , bytearray , VideoDecoder or dict) --
Dữ liệu được truyền dưới dạng đầu vào cho lệnh Video feature.0 với các trường "đường dẫn" và "byte"
Mã hóa ví dụ thành định dạng cho Arrow.

Flattendatasets.Video.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/video.py#L228[]
Nếu ở trạng thái có thể giải mã được, hãy trả về chính đối tượng địa lý đó, nếu không thì làm phẳng đối tượ

Pdfdatasets.Pdf

bộ dữ liệu lớp.Pdfdatasets.Pdfhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/pdf.py#L31[{"name": "decode", "val": ": bool = True"}, {"name": "id", "val": ": typing.Optional[str] = None"}]- mode (str , optional) --
Chế độ chuyển đổi pdf sang. Nếu Không có, chế độ gốc của pdf sẽ được sử dụng.

- decode (bool , defaults to True) --
  Có giải mã dữ liệu pdf hay không. Nếu sai ,
  returns the underlying dictionary in the format {"path": pdf_path, "bytes": pdf_bytes} .0

Thực nghiệm.
Tính năng Pdf để đọc tài liệu pdf từ tệp pdf.

Đầu vào: Tính năng Pdf chấp nhận làm đầu vào:

- A str : Absolute path to the pdf file (i.e. random access is allowed).
- A pathlib.Path : path to the pdf file (i.e. random access is allowed).
- Một lệnh với các phím:
    đường dẫn : Chuỗi có đường dẫn tương đối của tệp pdf trong kho lưu trữ dữ liệu.
    byte: Byte của tệp pdf.
      Điều này rất hữu ích cho các tập tin lưu trữ có quyền truy cập tuần tự.
- PDFplumber.pdf.PDF : đối tượng pdfplumber pdf.

Ví dụ:

```
>>> from datasets import Dataset, Pdf
>>> ds = Dataset.from_dict({"pdf": ["path/to/pdf/file.pdf"]}).cast_column("pdf", Pdf())
>>> ds.features["pdf"]
Pdf(decode=True, id=None)
>>> ds[0]["pdf"]
<pdfplumber.pdf.PDF object at 0x7f8a1c2d8f40>
>>> ds = ds.cast_column("pdf", Pdf(decode=False))
>>> ds[0]["pdf"]
{'bytes': None,
'path': 'path/to/pdf/file.pdf'}
```

cast_storagedatasets.Pdf.cast_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/pdf.py#L186[{"name": "storage", "val": ": typing.Union[pyarrow.lib.StringArray, pyarrow.lib.StructArray, pyarrow.lib.ListArray]"}]- storage
(Union[pa.StringArray, pa.StructArray, pa.ListArray]) --
Mảng PyArrow thành cast.0 pa.StructArray Mảng trong kiểu lưu trữ mũi tên Pdf, nghĩa là pa.struct({"bytes": pa.binary(), "path": pa.string()}) .
Truyền một mảng Mũi tên sang loại lưu trữ mũi tên Pdf.
Các loại Mũi tên có thể được chuyển đổi sang loại lưu trữ Pdf pyarrow là:

- pa.string() - it must contain the "path" data
- pa.binary() - it must contain the image bytes
- pa.struct({"bytes": pa.binary()})
- pa.struct({"path": pa.string()})
- pa.struct({"bytes": pa.binary(), "path": pa.string()}) - order doesn't matter

- pa.list(*) - it must contain the pdf array data

giải mã_exampledatasets.Pdf.decode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/pdf.py#L115[{"name": "value", "val": ": dict"}, {"name": "token_per_repo_id", "val": " = None"}]- value (str or dict) --
Một chuỗi có đường dẫn tệp pdf tuyệt đối, một từ điển có
phím:

- path : Chuỗi có đường dẫn file pdf tuyệt đối hoặc tương đối.
- byte: Các byte của file pdf.
- token_per_repo_id (dict , optional) --
  Để truy cập và giải mã các tập tin pdf từ kho lưu trữ riêng tư trên
  Hub, bạn có thể chuyển từ điển
  repo_id (str) -> token (bool or str).0 pdfplumber.pdf.PDF
  Giải mã tập tin pdf ví dụ thành dữ liệu pdf.

embed_storagedatasets.Pdf.embed_storagehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/pdf.py#L223[{"name": "storage", "val": ": StructArray"}, {"name": "token_per_repo_id", "val": " = None"}]- storage (pa.StructArray) --
Mảng PyArrow để embed.0 pa.StructArray Mảng trong kiểu lưu trữ mũi tên PDF, nghĩa là
pa.struct({"bytes": pa.binary(), "path": pa.string()}) .
Nhúng tệp PDF vào mảng Mũi tên.

Encode_exampledatasets.Pdf.encode_examplehttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/pdf.py#L80[{"name": "value", "val": ": typing.Union[str, bytes, bytearray, dict, ForwardRef('pdfplumber.pdf.PDF')]"}]- value (str , bytes , pdfplumber.pdf.PDF
or dict) --
Dữ liệu được truyền dưới dạng đầu vào cho lệnh Pdf feature.0 với các trường "đường dẫn" và "byte"
Mã hóa ví dụ thành định dạng cho Arrow.

Flattendatasets.Pdf.flattenhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/features/pdf.py#L173[]
Nếu ở trạng thái có thể giải mã được, hãy trả về chính đối tượng địa lý đó, nếu không thì làm phẳng đối tượ

Hệ thống tập tindatasets.filesystems.is_remote_filesythân cây

bộ dữ liệu.filesystems.is_remote_filesystemdatasets.filesystems.is_remote_filesystemhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/filesystems/__init__.py#L28[{"name": "fs", "val": ": AbstractFileSystem"}]- fs (fsspec.spec.AbstractFileSystem) --
An abstract super-class for pythonic file-systems, e.g. fsspec.filesystem('file') or s3fs.S3FileSystem .0

Kiểm tra xem fs có phải là hệ thống tập tin từ xa không.

Dấu vân taydatasets.fingerprint.Hasher

tập dữ liệu lớp.fingerprint.Hasherdatasets.fingerprint.Hasherhttps://github.com/huggingface/datasets/blob/4.2.0/src/datasets/fingerprint.py#L170[]
Hasher chấp nhận các đối tượng python làm đầu vào.