



Что делать, если моего датасета на нет на Hub?


Open in Colab Open Studio Lab

Вы знаете, как использовать [Hugging Face Hub](#) для скачивания датасетов, но часто складывается ситуация, когда нужные данные не хранятся у вас локально или на удаленном сервере. В этом разделе мы посмотрим, как библиотека  Datasets может быть использована для загрузки датасетов, которые не хранятся на Hugging Face Hub.

Loading a custom dataset

Копирува...



Посмотреть на  YouTube

Работа с локальными и удаленными датасетами

 Datasets предоставляет скрипты для загрузки собственных датасетов. Библиотека поддерживает несколько распространенных форматов:

Data format	Loading script	Example
CSV & TSV	csv	<code>load_dataset("csv", data_files="my_file.csv")</code>
Text files	text	<code>load_dataset("text", data_files="my_file.txt")</code>
JSON & JSON Lines	json	<code>load_dataset("json", data_files="my_file.jsonl")</code>
Pickled DataFrames	pandas	<code>load_dataset("pandas", data_files="my_dataframe.pkl")</code>

Как показано в таблице, для каждого формата мы должны задать тип скрипта загрузки в функции `load_dataset()` вместе с аргументом `data_files`, который указывает путь к одному или нескольким файлам. Начнем с загрузки набора данных из локальных файлов; позже мы увидим, как сделать то же самое с файлами, расположены на удаленном сервере.

Загрузка локального датасета

Для этого примера мы будем использовать датасет [SQuAD-it dataset](#). Это большой датасет для задачи question answering на итальянском языке.

Обучающая и тестовая часть расположены на GitHub, мы можем скачать файлы с помощью простой команды `wget`.


```
!wget https://github.com/crux82/squad-it/raw/master/SQuAD-it-train.json.gz
!wget https://github.com/crux82/squad-it/raw/master/SQuAD-it-test.json.gz
```

Выполнение этих команд запустит процесс скачивания файлов *SQuAD-it-train.json.gz* и *SQuAD-it-test.json.gz*, которые мы можем распаковать с помощью Linux команды `gzip`:

```
!gzip -dkv SQuAD-it-*.json.gz
```

```
SQuAD-it-test.json.gz:      87.4% -- replaced with SQuAD-it-test.json
SQuAD-it-train.json.gz:    82.2% -- replaced with SQuAD-it-train.json
```

После выполнения команд мы увидим, что архивы будут заменены файлами *SQuAD-it-train.json* и *SQuAD-it-test.json* в формате JSON.

 **Причина, по которой в примере выше перед командами расположен `!` заключается в том, что мы выполняем их в Jupyter notebook. Если вы хотите запустить эти команды в терминале – просто удалите `!`.**

Для загрузки JSON файла с помощью функции `load_dataset()` необходимо знать, с каким типом JSON-файла мы имеем дело: обычный JSON (похожий на вложенный словарь) или JSON, сформированный построчно. Как и многие датасеты для задач question-answering, SQuAD-it использует формат обычного JSON'a с текстом, хранящимся в поле `data`. Это означает, что мы можем подгрузить датасет, задав аргумент `field` следующим образом:

```
from datasets import load_dataset

squad_it_dataset = load_dataset("json", data_files="SQuAD-it-train.json", field="data")
```

По умолчанию при загрузке локальных файлов создается объект `DatasetDict` с меткой `train`. Мы можем изучить объект `squad_it_dataset`:

```
squad_it_dataset
```

```
DatasetDict({
  train: Dataset({
    features: ['title', 'paragraphs'],
    num_rows: 442
  })
})
```

Выше распечатана информация об объекте: число строк и колонки обучающего датасета. Мы можем посмотреть на один объект, проиндексировав его как `train` следующим образом:

```
squad_it_dataset["train"][0]
```



```
{
  "title": "Terremoto del Sichuan del 2008",
  "paragraphs": [
    {
      "context": "Il terremoto del Sichuan del 2008 o il terremoto...",
      "qas": [
        {
          "answers": [{"answer_start": 29, "text": "2008"}],
          "id": "56cdca7862d2951400fa6826",
          "question": "In quale anno si è verificato il terremoto nel Sichuan?",
        },
        ...
      ],
    },
    ...
  ],
}
```


Отлично! Мы загрузили наш первый датасет! Но пока мы это сделали только для обучающей части данных, хотя нам нужны и `train`, и `test` в одном `DatasetDict`, чтобы мы могли применить функцию `Dataset.map()` на оба подмножества сразу. Чтобы сделать это, мы можем передать в словарь в `data_files`. Сделать это можно так:

```
data_files = {"train": "SQuAD-it-train.json", "test": "SQuAD-it-test.json"}
squad_it_dataset = load_dataset("json", data_files=data_files, field="data")
squad_it_dataset
```

```
DatasetDict({
  train: Dataset({
    features: ['title', 'paragraphs'],
    num_rows: 442
  })
  test: Dataset({
    features: ['title', 'paragraphs'],
    num_rows: 48
  })
})
```

Это ровно то, чего мы хотели добиться! Далее мы можем применять различные приемы для препроцессинга данных: очистку, токенизацию и прочее.

 **Аргумент `data_files` функции `load_dataset()` очень гибкий и может являться путем к файлу, списком путей файлов или словарем, в котором указаны названия сплитов (обучающего и тестового) и пути к соответствующим файлам. Вы также можете найти все подходящие файлы в директории с использованием маски по правилам Unix-консоли (т.е. указать путь к директории и указать `data_files="*.json"` для конкретного сплита). Более подробно это изложено в [документации](#)  Datasets.**

Скрипты загрузки  Datasets также поддерживают автоматическую распаковку входных файлов, поэтому мы можем пропустить команду `gzip` просто передав в аргумент `data_files` пути к архивам:

```
data_files = {"train": "SQuAD-it-train.json.gz", "test": "SQuAD-it-test.json.gz"}
squad_it_dataset = load_dataset("json", data_files=data_files, field="data")
```

Это может быть полезно, если вы не хотите вручную разархивировать GZIP файлы. Автоматическое разархивирование также поддерживает распространенные форматы вроде ZIP и TAR, так что вы можете передавать и пути к таким файлам.


Теперь, когда вы знаете, как загрузить локально хранящиеся файлы, мы посмотрим, как подгрузить данные с удаленных серверов.

Загрузка файлов с удаленного сервера

Если вы работаете data scientist или программистом в компании, скорее всего ваши данные хранятся на сервере. К счастью, загрузка файлов с удаленных машин настолько же простая, насколько и загрузка их со локальной машины! Вместо пути к локальным файлам мы передаем аргументу `data_files` один или несколько URL, указывающих на нужные файлы. К примеру, датасет SQuAD-it расположен на GitHub, мы можем просто указать ссылку на файлы следующим образом:

```
url = "https://github.com/crux82/squad-it/raw/master/"
data_files = {
  "train": url + "SQuAD-it-train.json.gz",
  "test": url + "SQuAD-it-test.json.gz",
}
squad_it_dataset = load_dataset("json", data_files=data_files, field="data")
```

Эта операция вернет такой же `DatasetDict`, какой мы получали ранее, но избавит нас от загрузки и разархивирования файлов *SQuAD-it-*.json.gz* вручную. На этом мы завершаем наш обзор различных способов загрузки датасетов, которые не размещены на Hugging Face Hub. Теперь, когда у нас есть датасет, с которым можно поиграться, давайте погрузимся в различные методы обработки данных!

 **Попробуйте!** Выберите другой датасет, расположенный на GitHub или в архиве [UCI Machine Learning Repository](#) и попробуйте загрузить его с локальной машины и с удаленного сервера. В качестве бонуса попробуйте загрузить датасет в формате CSV или обычного текстового файла (см. детали по поддерживаемым форматам в [документации](#)).