Cache management

When you download a dataset from Hugging Face, the data are stored locally on your computer.
Files from Hugging Face are stored as usual in the `huggingface_hub` cache, which is at `~/.cache/huggingface/hub` by default.
See the Hub cache documentation for more details and how to change its location.

The Hub cache allows 🤗 Datasets to avoid re-downloading dataset files from Hugging Face every time you use them.

🤗 Datasets also has its own cache to store datasets converted in Arrow format (the format used by Dataset objects).

This guide focuses on the 🤗 Datasets cache and will show you how to:

- Change the cache directory.
- Control how a dataset is loaded from the cache.
- Clean up cache files in the directory.
- Enable or disable caching.

Cache directory

The default 🤗 Datasets cache directory is `~/.cache/huggingface/datasets`. Change the cache location by setting the shell environment variable, `HF_HOME` to another directory:

```
$ export HF_HOME="/path/to/another/directory/datasets"
```

Alternatively, you can set the `HF_DATASETS_CACHE` environment variable to control only the datasets-specific cache directory:

```
$ export HF_DATASETS_CACHE="/path/to/datasets_cache"
```

⚠ This only applies to files written by the `datasets` library (e.g., Arrow files and indices).
It does **not** affect files downloaded from the Hugging Face Hub (such as models, tokenizers, or

raw dataset sources), which are located in `~/.cache/huggingface/hub` by default and controlled separately via the `HF_HUB_CACHE` variable:

```
$ export HF_HUB_CACHE="/path/to/hub_cache"
```

💡 If you'd like to relocate all Hugging Face caches — including datasets and hub downloads — use the `HF_HOME` variable instead:

```
$ export HF_HOME="/path/to/cache_root"
```

This results in:

- datasets cache → `/path/to/cache_root/datasets`
- hub cache → `/path/to/cache_root/hub`

These distinctions are especially useful when working in shared environments or networked file systems (e.g., NFS).
See issue #7480 for discussion on how users encountered unexpected cache locations when `HF_HUB_CACHE` was not set alongside `HF_DATASETS_CACHE`.

When you load a dataset, you also have the option to change where the data is cached. Change the `cache_dir` parameter to the path you want:

```
>>> from datasets import load_dataset
>>> dataset = load_dataset('username/dataset', cache_dir="/path/to/another/directory/datasets")
```

Download mode

After you download a dataset, control how it is loaded by load_dataset() with the `download_mode` parameter. By default, 🤗 Datasets will reuse a dataset if it exists. But if you need the original dataset without any processing functions applied, re-download the files as shown below:

```
>>> from datasets import load_dataset
>>> dataset = load_dataset('rajpurkar/squad', download_mode='force_redownload')
```

Refer to DownloadMode for a full list of download modes.

Cache files

Clean up the Arrow cache files in the directory with Dataset.cleanup_cache_files():

```
# Returns the number of removed cache files
>>> dataset.cleanup_cache_files()
2
```

Enable or disable caching

If you're using a cached file locally, it will automatically reload the dataset with any previous transforms you applied to the dataset. Disable this behavior by setting the argument `load_from_cache_file=False` in Dataset.map():

```
>>> updated_dataset = small_dataset.map(add_prefix, load_from_cache_file=False)
```

In the example above, 🤗 Datasets will execute the function `add_prefix` over the entire dataset again instead of loading the dataset from its previous state.

Disable caching on a global scale with disable_caching():

```
>>> from datasets import disable_caching
>>> disable_caching()
```

When you disable caching, 🤗 Datasets will no longer reload cached files when applying transforms to datasets. Any transform you apply on your dataset will be need to be reapplied.

> [!TIP]
> If you want to reuse a dataset from scratch, try setting the `download_mode` parameter in load_dataset() instead.

Improve performance

Disabling the cache and copying the dataset in-memory will speed up dataset operations. There are two options for copying the dataset in-memory:

1. Set `datasets.config.IN_MEMORY_MAX_SIZE` to a nonzero value (in bytes) that fits in your RAM memory.
2. Set the environment variable `HF_DATASETS_IN_MEMORY_MAX_SIZE` to a nonzero value. Note that the first method takes higher precedence.