RelBench Documentation

Alessandra Cicciarelli

December 2025

1 Using RelBench to Download and Prepare Relational Datasets

RelBench provides a unified interface for working with realistic relational datasets. Each dataset is implemented through a `Dataset` class that includes a `make_db()` method responsible for downloading raw data (if needed), validating their integrity, and constructing a relational `Database` object consisting of multiple tables.

This section documents how to correctly use RelBench to download datasets, describes issues that may arise during the download and processing steps, and provides practical solutions.

1.1 Basic Usage

RelBench datasets are accessed through the `get_dataset()` utility:

```
from relbench.datasets import get_dataset

dataset = get_dataset("rel-amazon")   # or rel-stack, ...
db = dataset.make_db()
```

The call to `make_db()` performs three tasks:

1. Downloads the raw data if not already present in the cache.
2. Verifies file integrity (via SHA256 hashes when available).
3. Constructs a `Database` with normalized relational tables.

Once the database is built, its tables can be exported as CSV files:

```
for name, table in db.table_dict.items():
    table.df.to_csv(f"{name}.csv", index=False)
```

1.2 Download Mechanism and Caching

RelBench relies on the `pooch` library to download and cache raw files. The default download location can be inspected with:

```
import pooch
print(pooch.os_cache("pooch"))
```

If a file is already cached but corrupted or mismatched, clearing the cache is often necessary:

```
rm -rf <path returned by os_cache>/pooch/*
```

1.3 Common Issues and Solutions

1. Missing Packages

Some datasets require external libraries such as `tqdm` for download progress bars or `pyarrow` for JSON parsing. A common failure is:

```
ValueError: Missing package 'tqdm' required for progress bars.
```

Solution:

```
pip install tqdm pyarrow pooch
```

2. Broken or Outdated Dataset URLs

Some RelBench datasets (notably `rel-amazon`) originally relied on URLs hosted at:

`https://datarepo.eng.ucsd.edu/mcauley_group/data/amazon_v2/`

This domain is no longer active, which results in errors such as:

```
HTTPError: 404 Client Error: Not Found for URL
```

To fix this, the dataset wrapper must be updated to use the current location of the Amazon Review Data:

`https://mcauleylab.ucsd.edu/public_datasets/data/amazon_v2/`

3. Incorrect File Paths

Older versions of RelBench used directory names such as `reviewSets/`, which no longer exist in the updated dataset repository. The correct updated folders are:

- `metaFiles2/` for metadata
- `categoryFilesSmall/` for 5-core review data
- `categoryFiles/` for full review data

Ensuring that the dataset loader points to these locations resolves 404-Not-Found errors during retrieval.

4. SHA256 Integrity Check Failures

RelBench optionally validates file integrity using known SHA256 hashes:

```
ValueError: SHA256 hash of downloaded file [...]
does not match the known hash
```

This occurs when the upstream dataset updates its files, which is the case for several Amazon Review subsets. The downloaded file is deleted for safety.

Solution: Manually compute the new hash:

```
sha256sum meta_Books.json.gz
```

Replace the outdated hash in the RelBench loader, e.g.:

```
known_hashes = {
    "meta_Books.json.gz": "<new sha256>",
    "Books_5.json.gz": "<new sha256>"
}
```

Alternatively, set `known_hashes = {}` to disable integrity checks (not recommended for long-term reproducibility).

5. Cache Inconsistencies

If the local cache contains a partial or corrupted file, Pooch may attempt to reuse it, causing unexpected failures or parse errors.

Solution: Clear the cache and retry:

```
rm -rf ~/.cache/pooch/*
```

1.4 Validated Working Configuration for rel-amazon

A working configuration for `rel-amazon` in 2025 requires the following modifications:

1. Update the URL prefix to:

   `https://mcauleylab.ucsd.edu/public\_datasets/data/amazon\_v2`

2. Replace deprecated paths:

   - `reviewSets/` → `categoryFiles/`
   - `metaFiles/` → `metaFiles2/`

3. Update SHA256 hashes when upstream files change.

These modifications restore full functionality to the RelBench wrapper.