Movies Project Data Sources

Each of these is provided as a line-by-line JSON file, GZIP compressed, as can be read by Spark `spark.read.json` or Pandas.

```
data = pd.read_json('data.json.gz', orient='record', lines=True)
```

All of the files have a field `'imdb_id'` which is the IMDb label of the movie and can be used to identify across the files.

All of the data described here is on `gateways.sfucloud.ca` in the directory `/home/bigdata/movies.zip` (on the server, nothe cluster HDFS). You can download it (with any SFTP/SCP client) from there.

WikiData

The `wikidata-movies.json.gz` file contains information I thought might be relevant extracted from WikiData *(https://www.wikidata.org/)* : every movie I could identify, and every property that I thought might be useful for the project.

Each entry has the WikiData entity identifier, the WikiData label (≈title), and the English Wikipedia page title (possibly missing). Each of these properties from WikiData is included; most are lists of values because they could contain multiple values.

- based on *(https://www.wikidata.org/wiki/Property:P144)*
- cast member *(https://www.wikidata.org/wiki/Property:P161)*
- country of origin *(https://www.wikidata.org/wiki/Property:P495)*
- director *(https://www.wikidata.org/wiki/Property:P57)*
- filming location *(https://www.wikidata.org/wiki/Property:P915)*
- genres *(https://www.wikidata.org/wiki/Property:P136)*
- IMDb ID *(https://www.wikidata.org/wiki/Property:P345)*
- made profit? Boolean calculated from cost *(https://www.wikidata.org/wiki/Property:P2130)* and box office *(https://www.wikidata.org/wiki/Property:P2142)*
- main subject *(https://www.wikidata.org/wiki/Property:P921)*
- Metacritic ID *(https://www.wikidata.org/wiki/Property:P1712)*
- original language *(https://www.wikidata.org/wiki/Property:P364)*
- publication date *(https://www.wikidata.org/wiki/Property:P577)*
- Rotten Tomatoes ID *(https://www.wikidata.org/wiki/Property:P1258)*
- series *(https://www.wikidata.org/wiki/Property:P179)*

To see the WikiData description of these things, have a look at a popular movie with a lot of data: The Fellowship of the Ring *(https://www.wikidata.org/wiki/Q127367)* , Deadpool *(https://www.wikidata.org/wiki/Q19347291)* , etc.

Also included is the `genres.json.gz` file which is a mapping from WikiData entity identifiers to English genre names.

Rotten Tomatoes

The `rotten-tomatoes.json.gz` file contains all of the ratings I could get from Rotten Tomatoes *(https://www.rottentomatoes.com/)* (using the Rotten Tomatoes ID from the WikiData file).

The fields are:

- audience average rating (out of 5)
- audience percent who “liked it” (out of 100)
- audience ratings: the count of audience reviews
- critic average rating (out of 10)
- critic percent who gave a positive review (out of 100)

- IMDb ID
- Rotten Tomatoes ID

OMDb API

The `omdb-data.json.gz` file contains data from the OMDB API *(http://www.omdbapi.com/)* . The fields contain:

- IMDb ID
- awards won: text describing awards won by the movie
- genres: may or may not be similar to the WikiData genres
- plot summary: English text describing the plot

Methodology: How I got this data

You may or may not actually care about any of this. It is documentation for myself and available for you if you'd like to repeat the ETL, possibly adding data that you think will be useful. The code mentioned here is available in the `.zip` file described above.

Wikidata

I started with a wikidata.org JSON data dump *(https://www.wikidata.org/wiki/Wikidata:Database_download#JSON_dumps_(recommended))* , which is a nightmare of 24GB of BZIP2 compressed, deeply-nested JSON data.

It was disassembled into more usable files with `split-wikidata.sh` and copied to the cluster. The output of this is on the cluster HDFS at `/courses/datasets/wikidata`.

From there, `transform_download.py` turns the original mess into a data file that can be processed in a reasonable way. The output is about 1.7GB of `.parquet.gz` files that are actually vaguely manageable.

```
spark-submit transform_download.py /courses/datasets/wikidata wikidata-useful
```

The `build_wikidata_movies.py` Spark app further refines that into about 4MB of compressed JSON which contains facts actually relevant to the project. The output is what is provided as `wikidata-movies.json.gz`.

```
spark-submit build_wikidata_movies.py wikidata-useful-parquet
```

That program also produces the file mappings from WikiData IDs to human-readable labels for genres, `genres.json.gz`. It can produce a mapping of *all* WikiData IDs to labels, but that's commented-out.

Rotten Tomatoes

The Rotten Tomatoes data provided in `rotten-tomatoes.json.gz` was scraped from the web site. Web scraping is always a bad idea. Sometimes, it's the least bad idea available. I am not providing the code that does this.

OMDb API

The OMDb API *(http://www.omdbapi.com/)* provides a nice collection of data about movies, but it is rate-limited to 1000 requests per day.

The provided `get-omdb_data.py` fetched the data in `omdb-data.json.gz`. It requires the `build_wikidata_movies.py` output to get the list of movies (IMDb ID values, in particular). An API key must be provided on the command line:

```
python3 get_omdb_data.py 00000000
```

Because the API requires and API key and is rate limited, is cached to a DBM file, thus eliminating duplicate requests and allowing multiple days worth of fetches to be aggregated.

Updated Thu June 21 2018, 16:02 by ggbaker.