Administrator Guide to Database and Data Source

Kun-Lin Lee, Dennis/JIAJIE LIANG

November 15, 2015

Administrator Guide

TABLE OF CONTENTS

1.0 INTRODUCTION

1.1 Data Collection Overview

In today's world, buyers are being presented with an increasing range of choices while sellers are faced with the challenge of personalizing their advertising efforts. Recommendation engines help narrow choices to those that best meet your particular needs. The goal of such systems is to give user items or products that might interest him/her. Suggestions for books on Amazon, or movies on Netflix, are real world examples of recommender systems.

The most common approaches to recommendations are collaborative filtering and content filtering. Collaborative Filtering systems analyze historical interactions alone, while Content-based Filtering systems are based on profile attributes.

To support Collaborative Filtering and Content-based Filtering systems, given the proposal/advice from our sponsor, data augmentation begin with data source from MovieLens. To feed in the need of front-end OMDB, we augmented the data from , TMDB.

MovieLens, a researched-oriented recommender system, provides recommended movies to users based on user's volunteering rating on movies. It collects datasets about movies on the movielens.org, and provide the pre-processed dataset to public for research usage. We started from "MovieLens latest Datasets", with around 100,000 rating to 9,000 movies. After proof of concepts in algorithm design, we move forward to "MovieLens 20M Dataset", with around 20 million rating to 27,000 movies.

OMDB, short for open media database, is a film, TV Episode database for research project under CC Licence. They provide data through API to subscriber users. We augment with data source with OMDB data, with database provided directly by our sponsor.

TMDB, is originally started to help the movie, TV community with better collection of high quality posters, backdrop images and trailers links. We use this data source for front-end poster, movie metadata for content filtering, and as a similarity evaluation baseline.

1.2 Database Overview

2.0 DATA PREPARATION: DATA SCRAPING, PREPOSSESSING, MODELING

2.0.1 Data Sources From MovieLens, OMDB and TMDB

MovieLens provides the pre-processed csv format dataset to public on the http://grouplens.org/datasets/movielens/ [1], for research usage.

OMDB provides database for research project under CC Licence. OMDB database provided directly by our sponsor with csv format file.

TMDB, provides collection of high quality posters, backdrop images and trailers links. We use this data source for front-end poster, movie metadata for content filtering, and as a similarity evaluation baseline. The data collection starts from themoviedb.org through API requests/scraping.

2.1 TMDB SCRAPING AND PREPROCESSING

TMDB Scraping includes the following activities
- TMDB Dataset MovieLens "Movies" Scraping
- TMDB Trending Movies Scraping
- IMDB & TMDB Posters Resource Scraping

2.1.1 TMDB Dataset "Movies" Scraping and Preprocessing

TMDB has a Dataset called "Movies", which is the metadata for movies. The script TMDBMoviesScraping.py would initial requests to themoviedb.org API, which responses with JSON format string.

- The script takes the movieLens id from the OMDB 20 Million dataset links.csv
- map the movielens id to TMDB id
- send the request with TMDB id and other metadata (like API key)
- validate the response and encode correctly with string

(Encoding solution to this problem is costly to develop accurately, as it's more than choosing

a specific encoding schema, it's a overall recursively dictionary conversion into UTF-8, by a self-written parser)

Each JSON string of a movie includes the following information

Column Name	Explanation
poster_path	Poster path in TMDB database, access by: https://image.tmdb.org/t/p/original/poster_path
similar_movies TOP N	<= 20 movies with A movie ID. (TMDB Policy: 20)
backdrop_path	Backdrop image path in TMDB database, access by: https://image.tmdb.org/t/p/original/poster_path
budget	Movie budget: a numeric number
genres	a dictionary encoded with {id: genre name}
id	TMDB ID
imdb_id	IMDB ID
original_language	Original language
original_title	original title
overview	Movie Overview/Plot
popularity	Popularity index used only in TMDB, unverified source
adult	TRUE/FALSE, indicate allowing 18+ audience
production_companies	production companies id and name
production_countries	production countries id and name
release_date	a timestamp of year-month-day
revenue	movie revenue
runtime	By mintues, like, 96 means 96 minutes
spoken_languages	spoken languages
tagline	One or two lines summary of movie overview/plot

2.1.2 TMDB Trending Movies Scraping

TMDB has a collection of popular/top-rated/lastest/now playing “Movies”. The scripts TrendingMovies.py would initial requests to themoviedb.org API, which responses with JSON format string. Each JSON string of a movie includes the same information as above table. These information is used in the Ericsson similarity websites in the tab of “popular”, which display when the user haven’t searched for any preferred movies yet.

2.1.3 TMDB API Policy and Reference

TMDB API Key: 0e15eb439909c2e6a7ebbac73e049d34
Requests Rate Limit: 40 requests per 10 seconds[2]

Docs Reference http://docs.themoviedb.apiary.io/#reference

2.1.4 IMDB and TMDB Posters Resource Scraping

OMDB Dataset has a column as poster, pointing to an IMDB URL. The script JPGScrawler.py would would initial the IMDB URL, which responses with an JPG file.

- The script takes the movieLens id from the OMDB 20 Million dataset links.csv
- map the movielens id to IMDB id
- validate the response with HTTP response code check, if failed, retry for 3 times.
- Succeed, write to file. Failed after 3 trial, log down the information

For those with Error code in the IMDB URL, as our sponsor suggest, we look them up in the TMDB database. The script TMDB_JPGScrawler.py would would initial the IMDB URL, which responses with an JPG file.

- The script takes the movieLens id from the OMDB 20 Million dataset links.csv
- map the movielens id to TMDB id
- send the request with TMDB id and other metadata (like API key)
- validate the response with HTTP response code check, if failed, retry for 3 times.
- Succeed, write to file. Failed after 3 trial, log down the information

2.2 Data Model and exploratory analysis

2.2.1 MovieLens Data

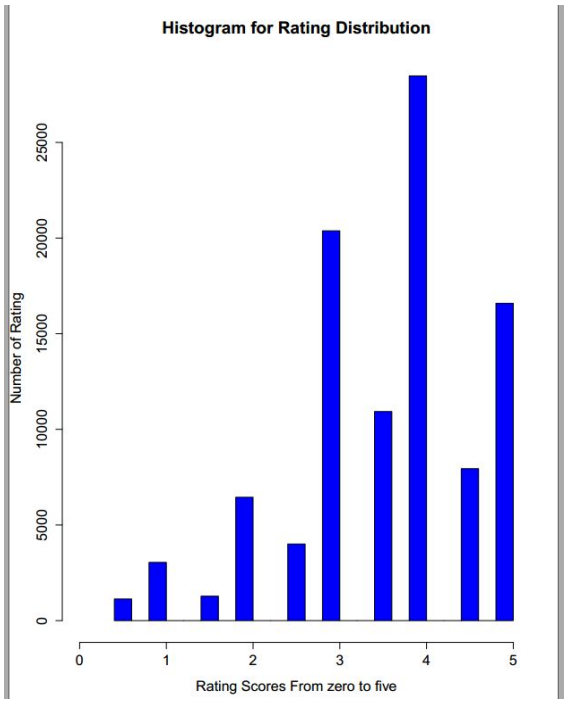
---

movieId - The ID of the movie: MovieLens users were selected at random for inclusion. Their ids have been anonymized. User ids are consistent between `ratings.csv` and `tags.csv` (i.e., the same id refers to the same user across the two files

title - Movie Title: Movie titles are entered manually or imported from <https://www.themoviedb.org/>, and include the year of release in parentheses. Errors and inconsistencies may exist in these titles.

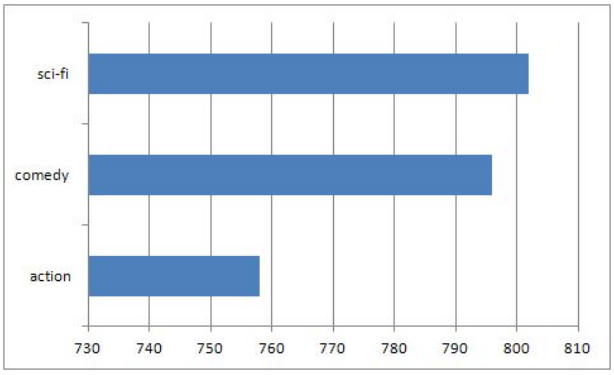
genres - Movie genres

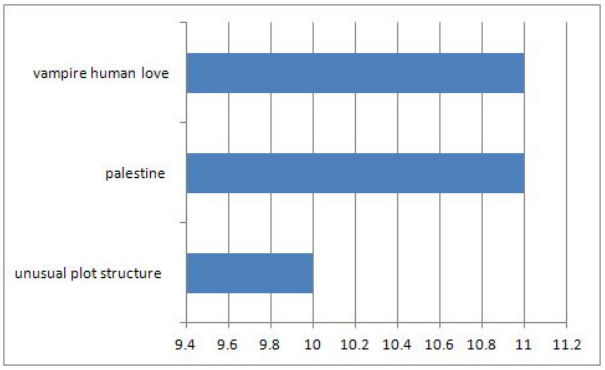
rating - The rating that this user gives this movie, Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars).



tag - The tag that this user gives this movie
(Tags are user-generated metadata about movies. Each tag is typically a single word or short phrase. The meaning, value, and purpose of a particular tag is determined by each user.)

Tag_popularity





2.2.1 OMDB Data

OMDB Table Description

Size After Cleaning	Records Numbers	Poster Availability	Episodes / Movies
915 MB	1,112,522	13.56%	episodes movie
Year	**Type**	**Sparsity**	**File Format After Cleaning**
Episodes, Movies: 1894, 1892	TV/Movie/Game	NA Data in String-based Field	CSV

Poster: URL-based content. 86.44% is unavailable.

Episodes / Movies: episodes includes TV series

Year: considering the released year, do we consider to recommend movie as old as 1892

2.2.3 TMDB Data

Posters

Field	Availability	SQL File

poster_path	90.4%, 24,682 /27,278	posteravail.sql (in github)

Budget

Field	Availability	SQL File
budget	22.1%, 6,039 / 27,278	budgetavail.sql (in github)

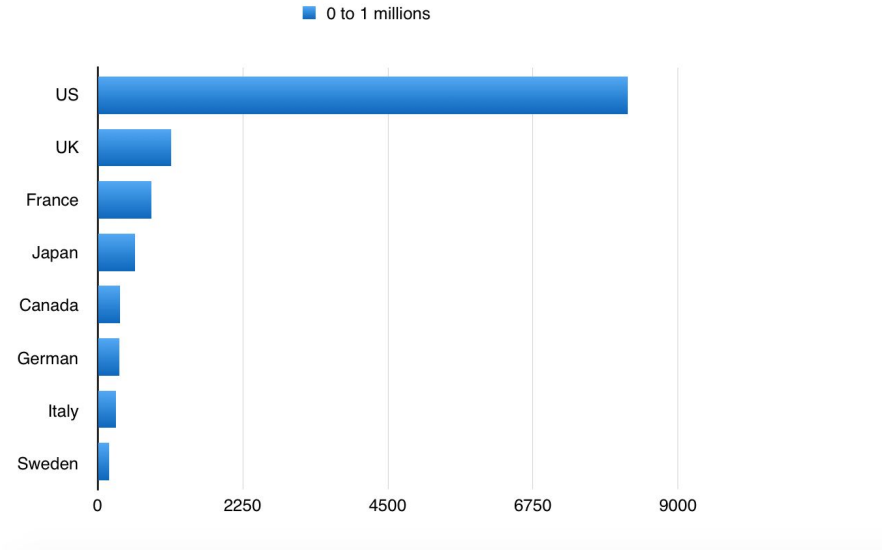
Overview

Field	Availability	SQL File
overview	90.05%, 24,708 / 27,278	OverviewAvail.sql

Revenue

Field	Availability	SQL File
revenue	19.56%, 5,335 / 27,278	RevenueAvail.sql (in github)

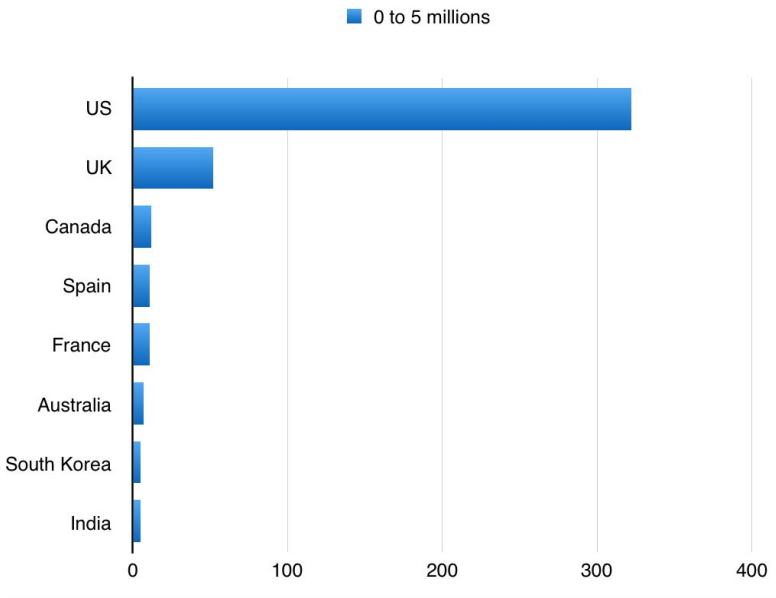
Revenue by production country, order by revenue between 0 to 1 millions



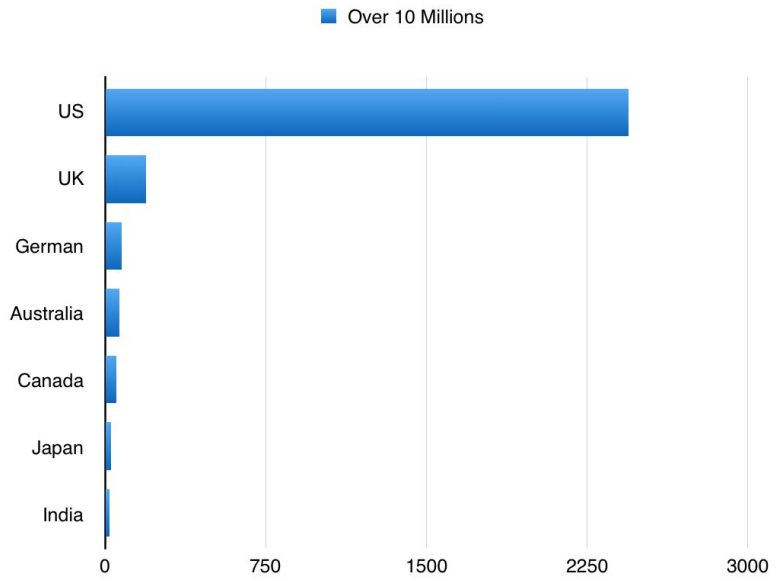
	0 to 1 millions
US	8221
UK	1138

France	829
Japan	580
Canada	347
German	338
Italy	282
Sweden	178

Revenue by production company, order by revenue between 0 to 5 millions



Revenue by production company, order by revenue over 10 millions



	Over 10 Millions
US	2442
UK	191
German	78
Australia	67
Canada	54
Japan	30
India	21

3.0 DATABASE

3.1 System Configuration

The system is ...

3.2

3.3

4.0 GETTING STARTED AND DEPLOYMENT STRATEGY

Extracting, Installing, and Running Scripts to download data and automated transferred to database.

4.0.1 Extracting from the .ZIP archive

In addition to user documentation, the .ZIP file contains mapping resource, and scripts for the web service. Both must be executed before python can be run. If it's desktop interface, unzip by any means you want. If it's in the terminal console, tyep in the following command, and then cd the destination folder to continue the scraping.

```
unzip delivery.zip -d [destination_folder]
```

4.1 TMDB Dataset "Movies" Scraping

4.1.1. Library Dependency, Code Structures & Resource

Prerequisite Resource

- links.csv, a file from 20 Millions dataset zip

Library Dependency

- pip
- pandas
- tmdbsimple

4.1.2. Execution Commands

1. install the dependency:

```
chmod +x libraryDependency.py
./libraryDependency.py
```

2. Download the Dataset Movies

```
chmod +x TMDBMoviesScraping.py
./TMDBMoviesScraping.py

Prompt: Choose the directory path you would like to store the dataset. e.g.
/Users/Ericsson/Documents/
```

```
[INFO] Enter your working directory full path:
/Users/dennis/Documents/Ericsson/
Your working directory is:  /Users/dennis/Documents/Ericsson/

Retrieving objects from themoviedb.org.................
[INFO] Retrieving latestest movie, Processing movie: That Ass In Yoga Pants 2
[INFO] Retrieving upcoming movie, Processing movie: Creed
[INFO] Retrieving upcoming movie, Processing movie: The Night Before
[INFO] Retrieving upcoming movie, Processing movie: Victor Frankenstein
[INFO] Retrieving upcoming movie, Processing movie: In the Heart of the Sea
[INFO] Retrieving upcoming movie, Processing movie: Krampus
```

Figure 1: Sample results after running ./TMDBMoviesScraping.py

3. Transfer to PostGreSQL database

```
chmod +x AutomatedTransferToPostgresql.py
./AutomatedTransferToPostgresql.py
```

```
Jiajies-MacBook:Ericsson dennis$ ./AutomatedTransferToPostgresql.py
[INFO] Data Transfer Automation start
[WARNING] Make sure your Postgresql database service open
[INFO]Transfer objects to database.................
10.91 KB / 10.91 KB [=============================================================================================] 100.00 % 0
7 rows imported into import.test
10.91 KB / 10.91 KB [=============================================================================================] 100.00 % 0
Jiajies-MacBook:Ericsson dennis$
```

Figure 2: Sample results after running ./AutomatedTransferToPostgresql.py

4.2 IMDB and TMDB Posters Resource Scraping

4.2.1. Library Dependency

Library Prerequisite

- pandas

4.2.2. Execution Command

1. install the dependency:

```
chmod +x libraryDependency.py
./libraryDependency.py

Command Prompt: Enter your Admin Password: [Mac: Apple password, Linux: Admin password]
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 11432  100 11432    0     0  17269      0 --:--:-- --:--:-- --:--:-- 17268
Extracting in /tmp/tmpMGfAIC
Now working in /tmp/tmpMGfAIC/setuptools-18.5
Installing Setuptools
running install
running bdist_egg
running egg_info
writing requirements to setuptools.egg-info/requires.txt
```

Figure 3: Download and install library

2. Download the Movies poster JPG from IMDB

```
chmod +x JPGScrawler.py
./JPGScrawler.py

Command Prompt: [INFO] Enter your working directory: e.g. /Users/Ericsoon/Documments
```

```
[INFO] Python script for JPG Start........
[INFO] Read in IMDB JPG PATH Mapping
[INFO] Enter your working directory full path:
/Users/dennis/Documents/Ericsson/
Your working directory is:  /Users/dennis/Documents/Ericsson/

Retrieving objects from IMDB..................
[INFO] Iteration:  1
[INFO] Downloading Poster for movielens id:  32898
[INFO] Iteration:  2
[INFO] Downloading Poster for movielens id:  71472
[INFO] Iteration:  3
[INFO] Downloading Poster for movielens id:  114376
```

Figure 3: running JPGScrawler.py

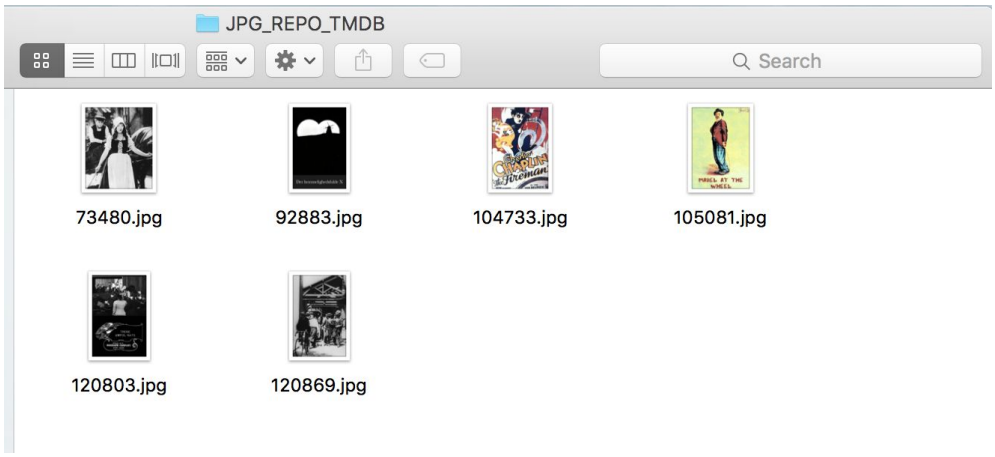


Figure 4: Downloading the images to Delivery/JPG_REPO

3. Download the Movies poster JPG (those which are missing in IMDB) from TMDB

```
chmod +x TMDB_JPGScrawler.py
./TMDB_JPGScrawler.py
```

```
[INFO] Python script for JPG Start........
[INFO] Read in TMDB JPG PATH Mapping
[INFO] Loading JSON files
[INFO] Iteration:  1
Index([u'movieId', u'imdbId', u'tmdbId'], dtype='object')
[INFO] Downloading Poster for movielens id:  120869
```

Figure 5: Downloading the images from TMDB

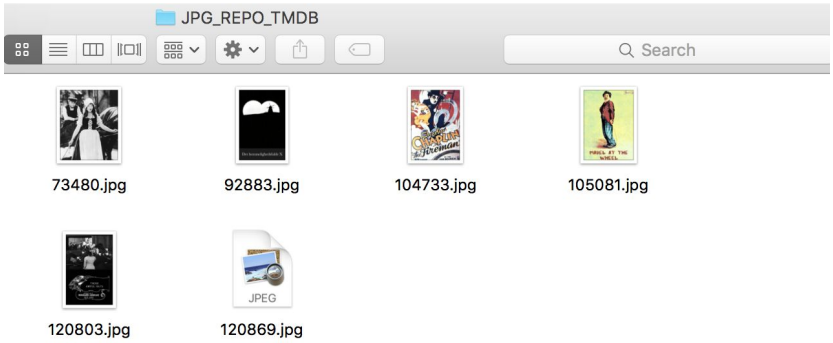


Figure 6: Image resides on Delivery/JPG_REPO_TMDB

4.3. TMDB Trending Movies Scraping

4.3.1. Library Dependency, Code Structures & Resource

Prerequisite Resource

- MovieLen links.csv, a file from 20 Millions dataset zip

Library Dependency

- pip
- pandas
- tmdbsimple

4.3.2. Execution Commands

1. install the dependency:

```
chmod +x libraryDependency.py
./libraryDependency.py
```

2. Download the Trending Movies

```
chmod +x TrendingMovies.py
./TrendingMovies.py
```

```
Out[17]: u'/Users/dennis/Documents/Ericsson/Delivery'

In [18]: %run TrendingMovies.py
[INFO] TMDB data retriever engine start
[INFO] Enter your working directory full path:
/User/dennis/Documents/Delivery/
Your working directory is:  /User/dennis/Documents/Delivery/

Retrieving objects from themoviedb.org..................
[INFO] Retrieving latestest movie, Processing movie: The Limehouse Golem
[INFO] Retrieving upcoming movie, Processing movie: Creed
[INFO] Retrieving upcoming movie, Processing movie: The Night Before
[INFO] Retrieving upcoming movie, Processing movie: Victor Frankenstein
[INFO] Retrieving upcoming movie, Processing movie: In the Heart of the Sea
[INFO] Retrieving upcoming movie, Processing movie: Krampus
```

Figure 7: Downloading Trending Movies

3. Transfer to PostGreSQL database

```
chmod +x AutomatedTransferToPostgresql.py
./AutomatedTransferToPostgresql.py
```

4.4 Database Deployment

5.0 REFERENCE

[1]: MovieLens Group Dataset API websites: http://grouplens.org/datasets/movielens/
[2]: Docs Reference http://docs.themoviedb.apiary.io/#reference