Team Koala Ninja (10)
Ankit Vij
Amanpreet Singh Saini
Joel Haynie

Class: CS 839 - Data Science Spring 2018
Prof: Dr. AnHai Doan
Date: 03/23/2018
RE: Project Stage 2

Entity Type: *Movies*

WebSites of Extraction:

http://www.imdb.com
http://www.rottentomatoes.com

Site Description:

Imdb - imdb.com is the “internet movie database”.

Rotten Tomatoes - rottentomatoes.com is a movie rating website that also holds metadata / attributes about the movies that are rated.

These were chosen because they are the internet's leading authority on movies and they hold much metadata / attributes about many movies.

Methods of Extraction:

The first step in extraction was to go to the websites, pick a few movie pages, and “take-a-look” at the html structure. Once the structure was ascertained we used BeautifulSoup and QT to make quick work of parsing and extracting the tags, tag attributes, and tag texts. This allowed us to define what information we were going to extract, and move onto finding the list of movie pages. We located a “search / listing” page, and the “take-a-look” strategy was once gain employed. This allowed us to figure out how to build the list of query URIs to get the ~3k movie pages to be fetched. Again BeautifulSoup and QT were used to extract the movie key words from a specific tags text attribute(href).

Once we had all the movie pages and data extracted we visited our schema and output format to put all the data into a usable form. We scanned through the table looking for & purging duplicate movie entities. Finally, we cleaned the data and put it into a similar format in preparation for the next stage.

Table Info & Schema:

We collected > 3000 tuples from each site across a variety of movie genres. We collected 3005 movies from rottenTomatoes and 3250 movies from imdb. The source of the data is not found in the data / schema but in the file name. The Tuples are of the form:

```
Url,Title,Score,Rating,Genre,Directed By,Written By,Box Office,Release
Date,Runtime,Studio
```

The description of the attributes is as below:

- **Url** - The movie URL
- **Title** - The title of the movie
- **Score** - The score given to the movie out of 10
- **Rating** - The rating assigned to the movie. Examples: PG-13, R, etc
- **Genre** - The genre the movie falls under separated by a semicolon(;). Examples: Action, Comedy, etc
- **Directed By** - The list of directors separated by a semicolon(;)
- **Written By** - The list of writers separated by a semicolon(;)
- **Box Office** - The amount of money made at the box office
- **Release Date** - The date of release of the movie
- **Runtime** - The duration of the movie
- **Studio** - The list of studios that produce the movie separated by a semicolon(;)

Tools Used:

- Development & Execution Environment:
  - PyCharm v 2017.3.3 (community edition)
- Libraries / Packages:
  - BeautifulSoup v 4.6.0
  - Requests v 2.18.4
  - PyQt v5.10.1

These libraries have many sub tool dependencies, but these are the major ones we used, these other dependencies were left out for brevity.

Beautiful Soup is a Python library for web scraping which provides methods to navigate and search a parse tree.

PyQt is set of Python bindings for the QT framework which are implemented as a set of Python modules. We used “QWebEnginePage” to give us an HTML blob for all the webpage code, including dynamic JS code, as it would have been rendered on a browser.

Conclusions & What we learned:

- Data rendered through JS can’t be directly parsed using Beautiful Soup. We needed to mimic the rendering to a browser using QT, which provides the final HTML as if it was rendered in a browser. We then passed this HTML to BeautifulSoup to extract the movie attributes.
- Scraping and cleaning data is hard work, and takes a lot of attention to detail.