839 Project Stage 2: Extracting Movie data from IMDB and Movie Numbers

Daniel K. Griffin
Department of Computer Science
University of Wisconsin Madison
dgriffin5@wisc.edu

Yudhister Satija
Department of Computer Science
University of Wisconsin Madison
ysatija@wisc.edu

Mitali Rawat
Department of Computer Science
University of Wisconsin Madison
mitali.rawat@wisc.edu

1 Introduction

For the Data Science 839 project stage 2, we were asked to develop rule based extractors to extract structured data from two overlapping sources of our own choosing, and to normalize both data sets to have the same schema. We decided to extract movies from the IMDB movies website, and the Movie Numbers website using a combination of standard python scripts, and the Scrapy framework for building web crawlers. A quick description of how we went about the task is given at a high level below, and the commands for how to run our web crawlers is provided in our root github README.md web page.

2 The Quick List

This section provides a quick reference sheet of our accomplishments, matching the list of what the .pdf file should contain on the course project web page. This section is meant to provide a quick synopsis of the required information to make evaluation easier.

1. **The Web data sources**: We used the following 2 sources:
   - IMDB movies list from http://www.imdb.com/list/ls032600534
   - Movie Numbers list from https://www.the-numbers.com/movies/#tab=letter
2. **How we extracted structured data**:
   - **Imdb extraction**: Extraction from the Imdb movie database website was a two stage process. The first stage was to parse data from the website using a Scrapy based web crawler, parse the data from the page using css based DOM selectors, and generate a .json data store of the parsed information. This first stage was fairly straightforward since the data was relatively flat, and there were no nested links that needed to be parsed. The second stage was to convert the .json data store into a .csv database type file.
   - **Movie numbers extraction**: Extraction from the Movie Numbers website followed the same two stage process as the Imdb extraction. The main difference was that extraction from this data source was more complicated, as nested links needed to be traversed at each page to extract all of the necessary movies and movie information and combine these for each movie in the callbacks of the responses to these pages. Also, our extractor for information from these pages relied on xpath based selectors, rather than css based selectors.We insert empty string in the csv file for missing values and for the attributes that are extracted for Imdb but not for Movie Numbers.

3. **Type of entity**: Our entities are movies. We have movie names and information like release date, directors and few cast member names. We also have MPAA rating of the movie and some figures on the sales.
4. **The two tables**: The two tables are included in the csv files:
   - "imdb.csv" for IMDB movie list
   - "thenumbers.csv" for Move Numbers movie list.

   The tables have the following identical schema:
   (title, year, mpaa, runtime, genres, star_rating, metascore_rating, description, director, stars, gross)
   The "stars" field refers to a few cast member names, separated by comma.
   The "genres" field refers to the genres the movie is categorized under, separated by comma.
   The "gross" field refers to sales numbers earned by the movie worldwide.
5. **Number of tuples**: We have obtained 4,291 tuples from the IMDB list. We have obtained 31,006 tuples from the Move Numbers' list.
6. **Open-source tools**:
   - **Python Scrapy**: Scrapy is an open-source python framework for quickly developing complex website scraping web spiders. Scrapy was designed to be easily integrated into a structured data parsing pipeline, and contains many integrated parsing modules for logging information, generating data statistics, generating emails given parsing events, and complex modules for parsing structured data using xpath and css DOM based selectors, web page link extraction tools, and support for integrating more complicated post-processing mechanisms for parsed data. We used Scrapy to develop web crawling spiders for both the imdb and movie numbers web sites, and use both the integrated xpath based and css DOM based selectors for pulling information from the structured web pages.

3 The Data Sources

We extracted movie information from the IMDB and Movie Numbers websites. IMDB is a general movie website that stores information about movies including categories of movies, related movies, movie ratings, movie descriptions, movie critiques, and many other pieces of information. For our purposes, we found a fairly substantial flat list of movies (about 4000 movies) structured in a browseable format for 100 movies per page, with links at the bottom of each page to move from one page to the next. Movie Numbers is another website that contains movie information, and holds roughly the same kinds of information as IMDB. The Movie Numbers website has an index of movies that allows a user to search for movies alphabetically based on movie name, and allows access to movies based off of a nested weblink indexing structure.

4 The Extractors

Both extractors are constructed of two smaller scripts. One script contains the code for a Scrapy based web crawler, and the other script contains the code for calling the crawler, and parsing its output into a .csv based file format. For the IMDB website, "IMDBSpider.py" contains the web crawler code, and "scrapeIMDB.py" contains the code that runs the IMDB spider and parses its output into a .csv file format. For the Movie Numbers website, "NumbersSpider.py" contains the web crawler code, and "scrapeMovieNumbers.py" contians the code that runs the Movie Numbers spider and parses its output into a .csv file format.

The "IMDBSpider.py" script is a fairly simple script that retrieves a page of moves, and uses css based DOM selectors to select each movie from the list of movies in the web page, and parses information from each movie. The overall structure of the crawler is fairly simple, as no nested links need to be traversed to get all of the movies from each web page. Also, since the web url follows a simple naming convention for all pages, the web crawler explores all web pages in parallel by expanding the number of every page for the web url, which speeds up parsing significantly. The "scrapeIMDB.py" web script calls the "IMDBSpider.py" script using a python subprocess to generate a .json file. When finished, the script reads the .json file, and uses python's built in json module to manipulate the json data, and write the json data into a csv based format.

The Movie Numbers extractor works almost exactly the same as the IMDB based extractor. The main difference however is in the complexity of the "NumbersSpider.py" web crawler. The Movie Numbers website uses a nested web link indexing structure. So, the website web crawler has to follow each nested link to find a movie information webpage before any information can be extracted. This means the overall web crawler was more complicated than the IMDB based crawler, as we were able to assume a flat indexing structure when parsing data from the IMDB based web pages.