Evan Hogan

Dr. Chris Teplovs

SI 330

23 April 2019

# The Dominant Medium

Movies allow us to see our favorite characters from literature come to life before our eyes. Instead of words on a page, viewers can now enjoy their favorite environments and characters on a screen in high-definition wonder. Well, sometimes at least. It's no secret that the movie industry does sometimes take an already well-written story and then makes a haphazard attempt to recreate it for a quick cash-grab. Nonetheless, I'm passionate about movies. I love to talk about my favorite movies, and even more so, defend my opinions about them. Debating becomes even more enjoyable once book-zealots enter the brawl. Therefore, I designed this project because I wanted to see, definitively, whether or not movies do their written counterparts justice or not.

As a quick foreword before we dive into the nature of my data sources, I would like to point out that yes, obviously the question I seek to answer, in its most basic form, is purely subjective on a person-by-person basis and is not empirically provable by any metric. However, public opinion is very quantifiable, and interesting to me. Also, my purpose for this project is not to prove whether or not books are better their movie adaptations, as the movie industry almost never makes a movie out of a sub-par book. Rather, my purpose is to prove whether or not public opinion shows that film adaptations of books do their source material any justice as far as being comparable in quality.

My first data source is a csv file of 10,000 popular books (https://github.com/zygmuntz/good books-10k/blob/master/books.csv). This dataset contains many different fields, such as, original title, a rating out of 5.00, publication year, number of votes, tags, etc. The second data source was public web API (www.omdbapi.com) that takes move title or TV series title strings as arguments and returns a json dictionary containing similar information: year, director, various rating metrics, cast, runtime, etc. However, the API is not perfect, and if given a title that doesn't exist, it will try its best to return the most relevant result.

The first thing that I did was import the omdb python package and used the set_default method with my API key so that all my API outputs would automatically be converted into python dictionaries. Due to the size of the project, I also had to pay the developer five dollars so that I could request more than 1,000 times per day. The next step was to create a list of strings from the 'title' column of the csv file that I would call using the OMDB API. For some reason, many book titles had parenthetical information after the book title itself, for example: Mockingjay (Hunger Games #3). I knew this would confuse the API, so I cleaned each title of the parentheticals using a regular expression and saving the clean titles into a list. I then passed each item from this list into the OMDB API, and if the title returned a movie dictionary with 'type' equal to 'movie' and 'runtime' greater than '60 min,' I added that titles into a new list that only contained book titles that returned feature length movies. This step took forever for the API to process, and I knew that next time I had to use the API, it would need to be my last. Now that I had a list of strings that returned movies, I passed each of those strings into the OMDB API once again and created a dictionary of my results. The dictionary contained the keys: 'query': the list of strings I used to call the API in order, ‘title’: the list of the title that each string returned, ‘year’: the year the movie came out, ‘RT’, ‘Metacritic’, and ‘IMDB’: the scores the movie got on three different publically acclaimed review outlets. In the event that any of these fields didn’t exist for a specific query string, I also included various try-except loops to catch errors and assign default values. I then created a dataframe from this dictionary, called movies_master.

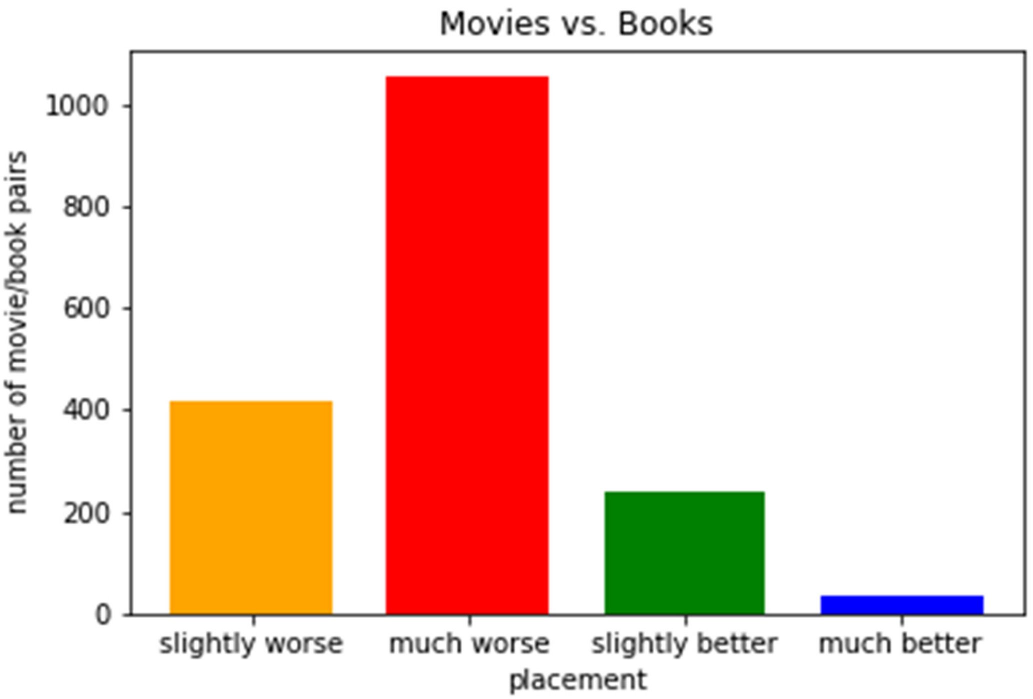
For my books dataframe, I selected the ‘title’, ‘original_publication_year,’ and ‘average_rating’ columns from the good books 10k csv file and saved that into a dataframe called ‘books_info.’ Again, I cleaned the titles of any parentheticals using the same regular expression. I then renamed the ‘query’ column in movies_master to ‘title’ and the ‘title’ column to ‘movie_title’ to make the upcoming merge easier. I merged the two frames on the ‘title’ column, which created a new dataframe of all the book info and movie info from book titles that I used to fetch movie data with. I called this new dataframe master_frame.

Now that I had a dataframe with all of the book and movie information for a given title string, I now had to get rid of any irrelevant or incorrect results using a series of various filtrations. As said before, the API will sometimes return incorrect data, especially with a short query string. For example, the book ‘Night,’ which is a Holocaust memoir written by a Romanian Jew, returned ‘Night at the Museum,’ a comedy featuring Ben Stiller. With this in mind, I filtered out rows if a book title of less than 9 characters didn’t exactly match the movie title. I also organized the table alphanumerically and eliminated all non-English results, which was only one row. I also filtered out rows if the movie year was younger than the book year, which would clearly indicate a flaw in the dataset. Most importantly, I also filtered out movies that did not return any data for all three of the ‘RT’, ‘Metacritc,’ or ‘IMDB’ fields, as I cannot make numerical comparisons between movies and books without movie data. I then converted each of the movie review scores into a number out of 100.00, rather than the percents, fractions, and out of 10.00 formats that they originally came out as from the API.

With filtration complete, I then had to decide how I was going to compare the books to the movies. I decided that I would average each of the 3 movie review scores into one as a separate field called ‘movie average.’ The first time I tried this, I was met with an error. This occurred due to the fact that when data was missing from any of the three movie score columns, I assigned the string ‘none’ as a default, which confused the **.**mean method I was using. After a quick fix from ‘none’ to Nan, I had my new column of average movie score. I then created another column of the difference between the average book score out of 100.00 and the average movie score out of 100.00. I called this column ‘difference.’ Now, I almost had all the data I needed to start my visualization. That is, before I came to a chilling realization; my dataframe had a few duplicate entries, apart from the average book rating field. I knew this had to do with books_info, one of the frames I used to merge. I found that different editions of the same book were rated differently. To rectify this error, I grouped by ‘title’ and took the mean of every field, which only affected the average rating for books, as that was the only field that was different across duplicates.

For the visualization, I decided to make a bar chart based on the differences between average ratings for books and movies. The method I came up with goes as follows. I created four new columns in my dataframe based on these differences: ‘slightly better,’ ‘slightly worse,’ ‘much better,’ and ‘much worse.’ The value for each column was a boolean, but as an integer, so either a 0 or 1 based on the value of the ‘difference’ column. For a movie to be slightly better, there had to be a difference less 0 and greater than -10. Much better, the difference needed to be -10 or lower. Slightly worse, the difference had to be between 0 and 10. Much worse, the difference had to be more than 10. The values for my bar chart were the sums of each boolean column. However, when I added these columns together, I was three short of the number of rows in my dataframe. Sure enough, there were three movies that were exactly comparable to its book counterpart. I decided to put these three into the ‘slightly better’ column by revising the category to be greater than or equal to 0. This did not skew the conclusions, as more than 1700 movie-book differences are shown in the graph. Here are my results.

To say I am disappointed would be an understatement. It seems that more often than not, film adaptations do not do their source material justice according to public opinion. Of the 1746 movie-book relationships, only 69 had a significant difference in favor of movies. Moreover, the sum of all other columns is still significantly less than the ‘much worse’ column. I never would have guessed that the results would be this definitive. I guess now the only option for me is to read more book in the future.


Movies vs. Books
number of movie/book pairs
slightly worse
much worse
slightly better
much better
placement