Box Office Movie Rating and Take Prediction

Mat Kelly, Sai Chaitanya Tirumerla, and Ibrahim Ben Mustafa
Department of Computer Science
Old Dominion University

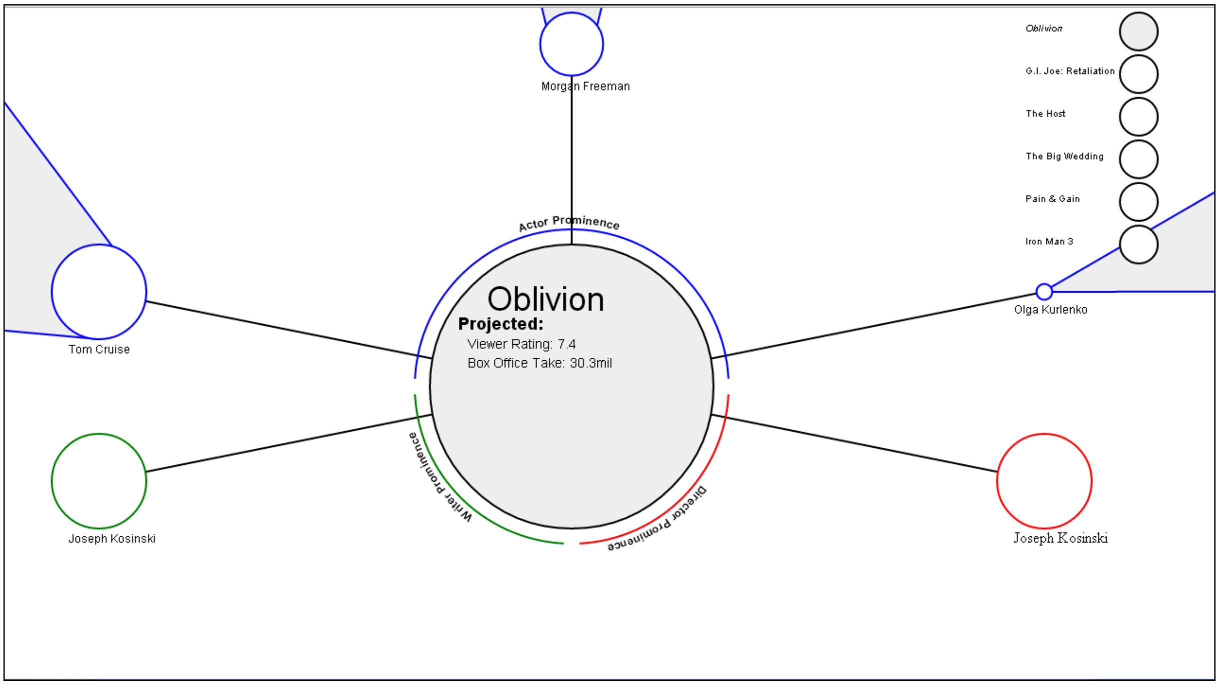

Fig. 1. The end-result visualization simplifies box office performance prediction into a simple black box system whose inputs are greatly reduced. By providing an exploratory interface, users are able to gauge the weight of each component (actor, writer, director) toward the end result through a node-and-link based navigational system.

Abstract—Predicting movies' box office performance is a bit like gambling and predicting the stock market: there's a lot of luck involved but analyzing the system's input frequently leads a better result. The inputs to box office prediction are infinite for the most part but there's little to argue that the actors involved in a new work and those actors' past performances play a huge role in a movie's success. Grasping the variables needed in the large data sets to make a box office prediction is an overwhelming task where a visualization would be useful. In this paper we describe a means we created to visualize these large data sets and distill them down into a simple causal approach that makes understanding the factors that affect box office performance easier.

1 Introduction

Predicting ticket sales and viewer ratings of movies at the box office[1] is a largely intractable, multivariate problem that requires an extensive ontology and some degree of luck. As the problem is complex, visualizing the factors that contribute to a movie's success helps to understand the weight that each parameter contributes. In this paper we described a visualization approach currently being developed to accomplish the task of visualizing the problem. Further, we describe a few approaches that can be taken to accomplish the feat to an acceptable degree.

2 Related Work

Ben Shneiderman coined what he called the "Visual Information Seeking Mantra" and noted that many successful products embodied the way of "Overview first, zoom and filter, then details-on-demand" in terms of design and interactivity [5]. Herman and Holten [1, 2] each went into details about graph-based navigation as well as the concerns of perspective and traversal. Raitner, in his PhD dissertation, formalized the structures that make up navigational graphs and optimized traversal methods [4]. As a portion of the data (see Section 4) required a sentiment classifier for Twitter tweets, we originally looked to the work of Pang and Lee, who evaluated various machine learning methods in attempting to classify sentiment [3]. We explored related works but were put to a halt when encountering Twitter issues, which limited our data source, detailed in Section 4.

3 Approach/Design

As the problem of movie prediction is multivariate, performing dimensionality reduction allows us to take control of the problem. Much like stock market prediction, the number of variables that affect the outcome is largely infinite and stochastic, which allows us to develop a model to exploit the variance caused by randomness while harnessing the causal nature of a movies' attributes to its box office take and rating.

We considered adopting an intentionally naïve approach of ignoring many parameters for the movie's success with our design and focused on the following contributory inputs:

- *Mat Kelly, Sai Chaitanya Tirumerla, and Ibrahim Ben Mustafa are graduate students of computer science at Old Dominion University, e-mail: {mkelly, stirumer, iben}@cs.odu.edu*
- *Live visualization:* `http://www.cs.odu.edu/~mkelly/semester/2013_spring/project2/`

[1] `http://boxofficevast.org/vast-welcome.html`

- The top three actors in the movie
- The leading writer of the movie
- The leading director of the movie
- Each of the previously listed member's previous success with movies (rating and box office take)

3.1 Strategic Data Collection and Cleaning

We were given a very limited corpus of movies to focus on yet the algorithm should be generally applicable as long as the attributes of each movie to be analyzed are present. From our data set (see Section 4) we pivoted on the movies provided and proceeded with the data collection phase. For each movie we obtained the first three actors. For each actor, we obtained the sub-corpus of movies with which the respective actor was involved. We trimmed this sub-corpus to only movies in which the respective actor was billed in the first three actors on the respective movie's list of actors. The purpose of this was to reduce the dimensionality to the degree that would be useful for our heuristic and no further. For each movie that each actor was in (as limited by the above), we repeated the process of selecting the first three actors and performed the operation recursively. As the corpus is progressively built, data about each movie is retained including the critics' rating of the movie (on a scale from 0.0 to 10.0) and the box office take on the opening weekend of the movie (in U.S. dollar, or converted if in another currency).

3.2 Visualization Design

We wished to organize the information that we chose as a focus in a simple clean way that allows the user to explore each parameter to our prediction as well as how each parameter contributed to the Rating and Take results. Our initial mockup (Figure 2) added each of the elements listed above in a clean, dynamic form, hiding parameters we believed were beyond the scope or not displaying them in the interface thereby implying their lack of contribution toward our prediction.

With movies having a natural precedence of the people involved in their creation, we considered this through the sub-corpuses created to generate a degree of prominence that an actor's inclusion provided while maintaining the display of billing order. Though it is likely that billing order (the relative position in the credits where an actor's name resides) closely correlates to prominence, we maintained precedence to ensure inclusion of the actor (recall, we only consider the top three actors) in the output regardless of prior work.

The user interface elements are node and link based. In the center contains the movie (**Movie Circle**) in focus by the user with its title shown (top half of circle) as well as our prediction results (bottom half of circle), including Viewer Rating and Box Office Take (Figure 2). On the perimeter of the top half of the circle is a clockwise scale of inverse Actor Prominence, as we have computed, i.e., the most prominent actors relative to the movie are on the left.

Along the top half of the circle are attached three nodes (**Actor Nodes**), each with a link, symbolizing each of the three contributing actors (per above). The radius of each node signifies the actor's impact toward the success of the film. Emanating from each Actor Node toward the closest border of the visualization is a colored area (**Actor Swath**). The size of each Actor Swath is relative to the calculated prominence of the actor from past films within the accumulated corpus with the width of the swath of each actor, relative to the other actors' Actor Swaths at the same distance from the respective Actor Node being representative of the relative cardinality of the impact of the respective actor's previous films.

Attached to the bottom half of the Actor Node are two categories of contributory inputs, the writers and the directors. For simplicity, we only consider the lead of each but make the secondary of each available in the user interface for navigational and exploratory purposes. As shown on Figures 1 and 2, these other actors are displayed as smaller nodes to symbolize their lack of importance in the computation.

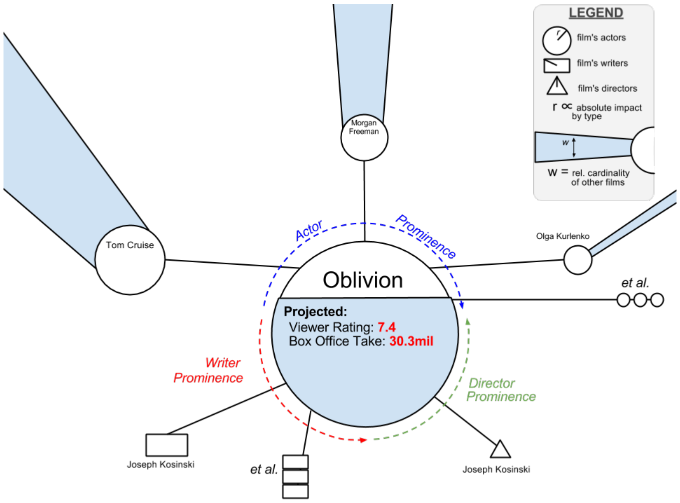

Fig. 2. The original mockup (compare to the end-result in Figure 1) contained all the elements we wished to contain in the visualization through a linked-node-like system. The swaths connected to each actor node represent Shneiderman's details-on-demand [5] while various other information visualization tactics were instilled.

3.3 Rating and Box Office Take Estimation

Our initial approach for estimating the box office take and rating was simple yet effective. We first analyzed our estimation result on a movie that already has these outputs associated with it. We first considered the average rating of the collection of movies in which an actor was billed in the first three actors. For example, a movie's rating with three actors (A_1, A_2, A_3) can be naïvely estimated as the average of the three actors' each respective movie rating average. So if A_1's average for all movies in which he was billed in the top 3 is *4*, it follows from the values in Equation 3 that the estimated rating for the collaborative movie would be $\frac{4.0+8.0+8.5}{3} = 6.8$. Likewise, the unweighted contribution of each actors' respective previous works (again, being limited to where they were billed in the top 3) is shown in the same figure. The variability of prediction comes on how to weight these values. We took an initial approach of simply weighting the values based on the number of movies for each actor over the sum of the movies of all three actors.

$$\begin{aligned} A_{1R} &= 4.0 \quad & A_{1\$} &= \frac{20\ Million}{\#\ of\ movies\ A_1} = \frac{20}{3} \\ A_{2R} &= 8.0 \quad & A_{2\$} &= \frac{30\ Million}{\#\ of\ movies\ A_2} = \frac{30}{4} \\ A_{3R} &= 8.5 \quad & A_{3\$} &= \frac{25\ Million}{\#\ of\ movies\ A_3} = \frac{25}{1} \end{aligned}$$

Fig. 3. The average contribution toward the rating of a movie by the collection of the top 3 actors in the new movie can be calculated by considering each actor's average box office take.

$$\left(\frac{3}{8}\right)\left(\frac{20}{3}\right) + \left(\frac{4}{8}\right)\left(\frac{30}{4}\right) + \left(\frac{1}{8}\right)\left(\frac{25}{1}\right) = 9.3$$

Fig. 4. When a simple cardinality based weighting system is used, a reasonable result occurs.

4 The Data

The data supplied was from two collections, Twitter[2] tweets and IMDB database dumps in the form of .list files. The Twitter data consisted of a collection of IDs, encoded in XML, that required querying the Twitter API to obtain the contents. The content of the Tweets at

[2] http://twitter.com

the IDs, presumably, were about the respective movie with which the data set associated to the organized XML files.

We ran into issues beyond our control for our first attempt at data cleaning. We tried integrating with both tweepy[3] and then the python-twitter[4] Python libraries as recommended by Twitter's official developer documentation[5]. After some frustration (the documentation is not very clear), we were able to interface and pull down tweets using the API. Anonymous access to the tweets by ID could be obtained but was limited to 150 tweets per "time-frame", a variable Twitter adjusts. At the time of our data acquisition, this variable was 15 minutes, leaving us to believe that anonymous access for data acquisition was not temporally feasible. We opted to go the OAuth 1.1[6] route, which required obtaining API access token, which we did. After running the tweet acquisition program for longer we noticed it failing more frequently. Upon investigating the issue, we saw in the documentation that authenticated access (versus anonymous access) to the API was limited to 350 queries per time-frame and is attached to the API key rather than the IP address, as was done with anonymous access. In what little data we did obtain, it did not seem to be useful for what we wanted to accomplish without intense lexical analysis and still then, based on Pang's work [3], the results using standard machine learning schemes might not yield a sufficient result. As the data did not appear as useful as the secondary data source, we opted for the latter.

The second data set consisted of many text files separated with movie details and published weekly[7] by IMDB. With the information we wanted for a basis being actors, movies, ratings and box office take, we utilized the actors, business, movies and ratings ".list" files in the IMDB repository. These files were very large (737MB, 42.6MB, 112MB, and 29.1MB, respectively) and many text editors were unable to even open the files for inspection. For some files, we were able to use the Sublime Text editor[8] after the likes of many others failed due to the memory footprint. In writing the Python code to handle these text files, we had to be memory conscious and first ensured that Python's `file.readline()` removed previous lines from RAM once they were no longer used.

This data set proved unwieldy and difficult to establish links through programmatic processing. We opted instead to manually extract the data needed to create the hierarchy between actors, movies, rankings and box office take. After manually obtaining the data, an aggregate of the IMDB .list data set, data from the IMDB official API, and data derived from queries to the IMDBAPI.org API, we manually sanitized it then ran it through a Python-based processing script that we had created to get the data into a form readable by JavaScript, namely JSON (Figure 5).

5 Implementation/Tools

The graphical and interactive portion of the visualization was built using D3.js 3.1.6[9], jQuery 2.0.0[10] and glue Javascript. We performed data cleaning by scripting in Python[11]. The IMDB API was queried with Python (after encountering the JavaScript Same Origin policy restriction) and the IMDBAPI.org (an unofficial partial wrapper to the IMDB API) API was queried with JavaScript. The IMDbPY[12] library was evaluated and initially used but we found compatibility inconsistent and resorted to manual data cleaning and acquisition only after spending more than the amount of time fiddling with the package than would have taken to manually extract the data.

```
G.I. Joe: Retaliation (2013)
6.2
Dwayne Johnson (nm0425005)
Jonathan Pryce (nm0000596)
Byung-hun Lee  (nm0496932)

Snitch (2013)
6.8
Dwayne Johnson (nm0425005)
Barry Pepper (nm0001608)
Jon Bernthal (nm1256532)

Masquerade (2012)
7.6
Byung-hun Lee  (nm0496932)
Seung-yong Ryoo (nm2440627)
Hyo-ju Han    (nm2174122)

nm0425005
Dwayne Johnson
Pain & Gain (2013): tt1980209  $20,244,505 - 7.0
G.I. Joe: Retaliation (2013) : tt1583421  $40,501,814 - 6.2
Snitch (2013) : tt0882977 $13,167,607 - 6.8
Journey 2: The Mysterious Island (2012) : tt1397514  $27,335,363 - 5.7

nm0000596
Jonathan Pryce
Dark Blood (2012) : tt0293069  N/A - N/A
Hysteria (2011)   : tt1435513  $35,656 - 6.7
My Zinc Bed (TV 2008) :tt1056101  N/A - 5.6
Sherlock Holmes and the Baker Street Irregulars (TV 2007) : tt0892743  N/A - 6.3
The Moon and the Stars (2007) : tt0460873  N/A - 6.1
Brothers of the Head (2005) : tt0432260 $10,794 - 6.3
De-Lovely (2004) : tt0352277 $123,920 - 6.4
The Affair of the Necklace (2001) : tt0242252 $125,523 - 6.0
```

Fig. 5. We manually extracted data and post-processed it into a form that was consumable by JavaScript and D3. Shown here is an excep of the post-processed data.

6 Intended Usage

Though incomplete due to time constraints, we will note here a sample use case to aim toward for future work. Our original intention with the swaths (Figure 1) was to allow the user to view the actor's other works that serve in the calculation of the currently viewed movie's predictions. A user might click on the swath and either the visualization would pan left or expand to accommodate the additional linked data. Another addition would be for the user to obtain addition data about the movies or the actors by clicking on the nodes. As is implemented but not described elsewhere in this document, a "movie selector" functionality exists at the top right of the visualization that is a single disconnected node that, when clicked, expands into the view shown in Figure 1. When a user selects another movie available, the visualization's canvas is cleared and the new movie is loaded. This could be expanded in future work to allow a similar functionality but to show only the other works in which a "selected" actor was involved. This might reside in the actor-centric context as opposed to the movie-centric context shown.

7 Conclusions

Creating this visualization was largely an exercise in data cleaning and attempting to link data from very large data sources. We believe that excluding the Twitter data set is effectively noise reduction and would not reduce but rather, improve the likelihood of prediction accuracy for box office rating and take. Our prediction algorithm is still it infancy and we hope to improve on it in future by investigating more of what others have done in terms of box office prediction algorithms, which we neglected to do for this visualization.

8 Caveats

In the interest of time, we excluded some key factors that might have contributed to a better prediction system.

- We only conisdered the 5 most recent movies by the actor to minimize the temporal complexity of the calculation. This likely affected actors who had many contributions where only the recent ones have flopped (e.g., Steve Buschemi).

- We excluded short films, foreign films, and movies whose initial release was not in the United States (e.g., Ripley Under Ground, Barry Pepper), as IMDB frequently did not contain the required information necessary to perform the prediction.

[3] https://github.com/tweepy/tweepy

[4] https://github.com/bear/python-twitter

[5] https://dev.twitter.com/docs/twitter-libraries

[6] https://dev.twitter.com/docs/auth/using-oauth

[7] http://www.imdb.com/interfaces

[8] http://www.sublimetext.com/

[9] http://d3js.org/

[10] http://jquery.com/

[11] http://www.python.org/

[12] http://imdbpy.sourceforge.net/

- We modified the shape of the directors and writers nodes from the mockup for simplicity, as circles were more consistent and easier to draw than polygons on an svg canvas.

- We derived our corpus from the given set of movies and recursed from there, making it likely that new movies without all of the movies being in our corpus would require the corpus to be augmented prior to giving a prediction.

- We converted foreign currencies to United States dollars for consistency at the rate of the movie's release. This likely had subtle inaccuracies.

9 Final Thoughts

Unlike the first project for the course, we had learned to not work backward from the visualization to the data. Data acquisition was the greatest stumbling block due to restrictions from IMDB and Twitter and the unweidly and lengthy IMDB list files. To counter this, we looked to IMDBAPI.org, which provided a querying interface to access data but the data returned was not everything needed. We resorted to extensive data acquisition only after having futzed with the Twitter, IMDB API, IMDBAPI.org API and the IMDB list files. We, again, like project one, ran into time constraints due to the unforeseen complexity of the data. In terms on individual contributions, Mat designed and coded the visualization, performed all of the programming and created the report. Chaitu fetched the IMDB data, created the presentation and documented the initial experience with the IMDB API. Ibrahim's contributions were to do the extensive laborious manual data fetching and exploring the IMDbPY package.

References

[1] Herman, I and Melancon, G. and Marshall, M.S. Graph Visualization and Navigation in Information Visualization A Survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43.

[2] Holten, Danny. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, (5):741–748, 2006.

[3] Pang,B. and Lee, L. and Vaithyanathan, S. Thumbs Up? Sentiment Classification Using Machine Learning techniques. In *Proceedings of the Confernce on Empirical Ethods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

[4] Raitner, M. *Efficient Visual Navigation of Hierarchically Structured Graphs*. PhD thesis, University of Passau, 2004.

[5] B. Shneiderman. A Grander Goals: A Thousand-Fold Increase in Human Capabilities. *Educom Review*, (32):4–10, November 1997.