MOOVIES

by Team NEMO (13)

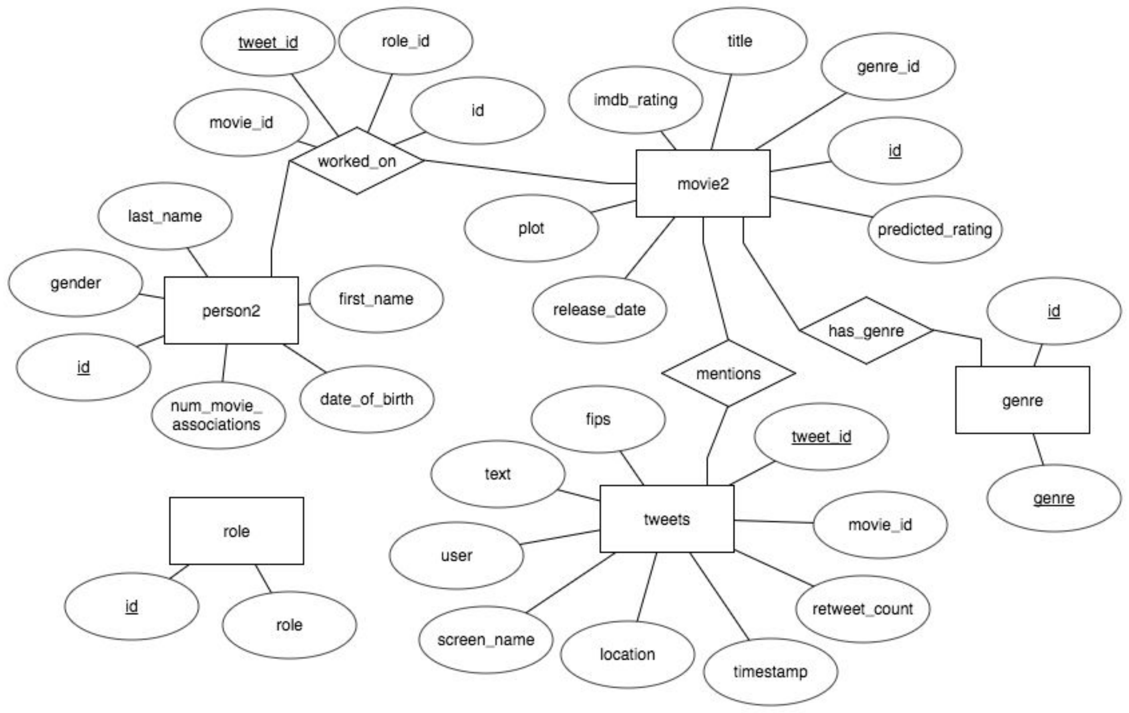
Nga Thai, Edward Moore, Max Li, Omar Abdulaev

1. Project Idea

MOOVIES is a website that allows users see rating predictions for upcoming movies. MOOVIES also analyzes and integrates tweets and data from Twitter to help users gauge public interest. MOOVIES is for anybody who is tired of IMDb and wants to see everything they want to see in one simple, well-organized page.

2. ER Design and Schema Design

ER Design



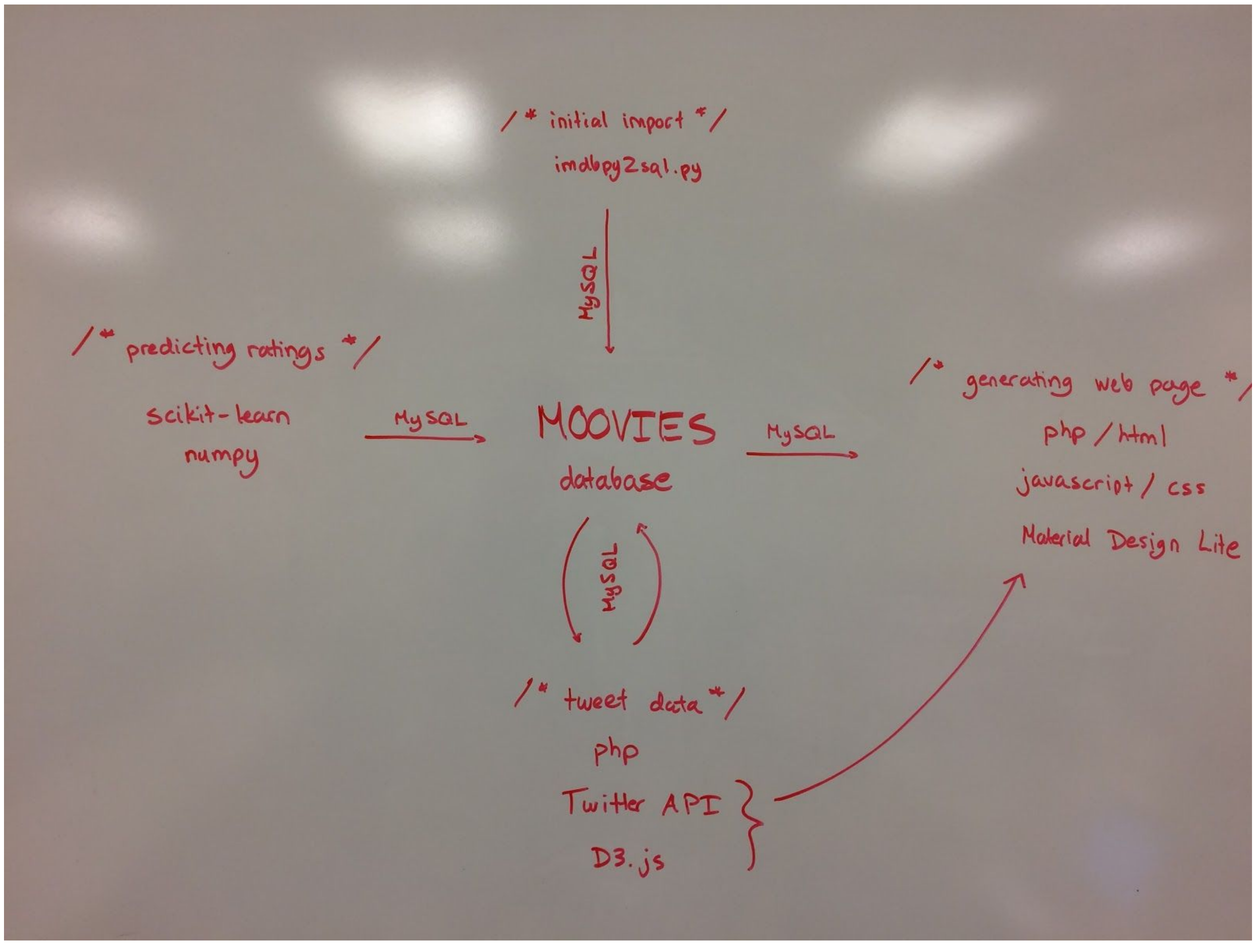
Schema Design

- genre(id, genre)
- movie2(id, title, plot, release_date, genres, predicted_rating, imdb_rating)
- person2(id, first_name, last_name, gender, dob, num_movie_associations)
- role(id, role)
- tweets(movie_id, tweet_id, text, location, fips, timestamp, user, screen_name, retweet_count, favorite_count, top_tweet_count)
- worked_on(person_id, role_id, movie_id, id)

3. Data Import

- We imported the IMDb database using IMDbPY and then used SQL queries to retrieve the appropriate information for use in our own Database.
    - For example, we used the query:
      "INSERT INTO moovies.person2(name, id) (SELECT DISTINCT imdb.name.name, imdb.name.id FROM imdb.name, imdb.cast_info WHERE (role_id = 1 OR role_id = 2 OR role_id = 8) AND imdb.cast_info.person_id = imdb.name.id)"
      to put in the name and ids of the people who were actors, directors, or actresses in our person2 table
- We inserted the twitter data by using the twitter API to get the information we needed about each tweet associated with a movie and sql statements
- Final State:
    - genre - 34 rows, 2 attributes
        - clean
    - movie2 - 1150554 rows, 8 attributes
        - somewhat clean, missing certain information from certain categories such as imdb_rating and plot
    - person2 - 4056682 rows, 6 attributes,
        - somewhat clean, some rows do not have a date of birth
    - role - 12 rows, 2 attributes
        - clean
    - tweets - 7208 rows, 11 attributes
        - mostly clean, some rows (not very many) are missing location
    - worked_on - 14387634 rows, 4 attributes
        - clean

4. Development Environment



- PHP, CSS, HTML were used for the website design and basic functionality.
- The material design light library was for the appearance of several aspects of the website: font, search bar, search result tables, radio buttons.
- We chose PHP because it's straightforward and relatively easy to use, especially on a virtual machine, while CSS and HTML are necessary for any website. The material design light library was chosen mainly because we liked its appearance.
- Used Twitter API to display top tweets and get the number of tweets about a movie from each state and map it
    - We used the Twitter API because twitter is a very popular form of social media so the user can gauge interests/make an opinion about a movie and decide whether to see it or not based on what other people said/are currently saying about a movie and make our application seem more visual
    - It was an okay experience because the documentation wasn't very good so we struggled a little at first to figure out what information corresponded to what we needed about a tweet in order to display it
- Used D3 to display our map of tweets per state about a movie

- ○ We used D3 because the map functionalities were pretty cool like coloring a state based on how many tweets came from people in the state and the ability to zoom and make our application seem more visual and interactive
- ○ Using D3 was pretty easy but the time consuming/confusing part was figuring out what permissions we needed to set in order to write data to a csv file since root permission was needed and how to set these permissions

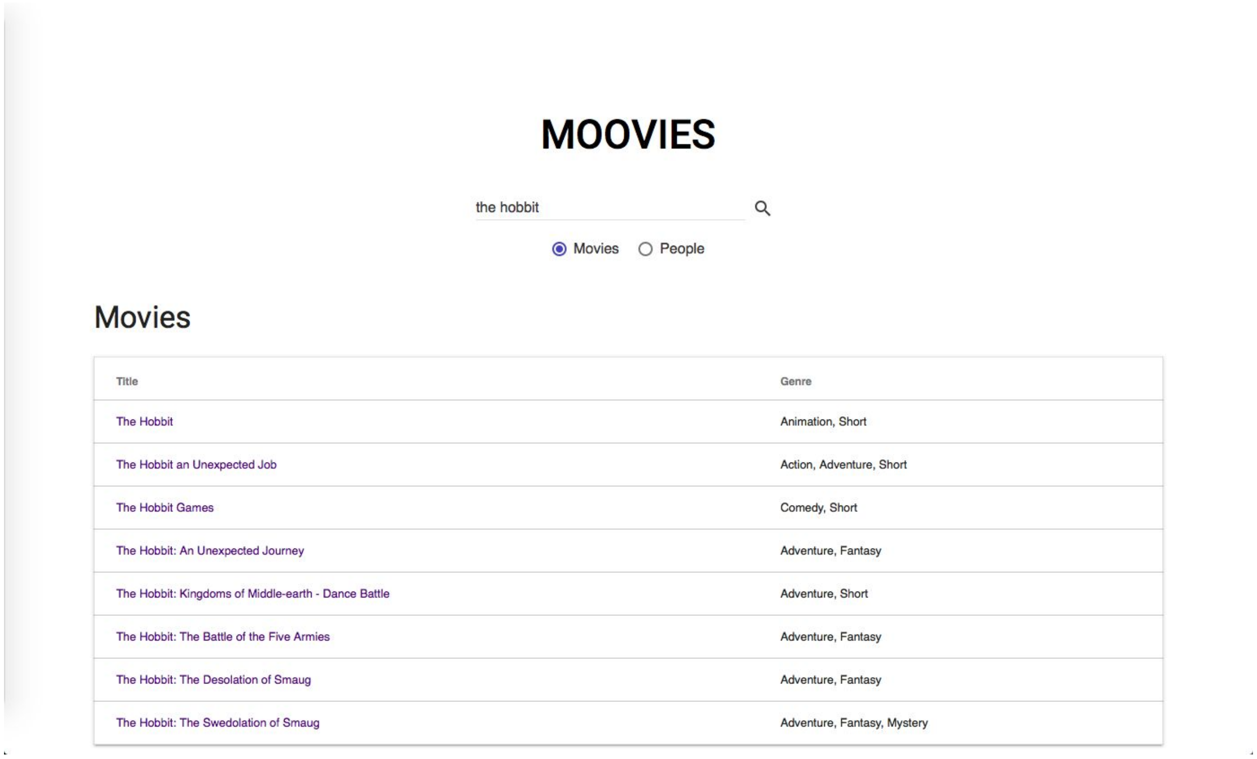
- We used scikit-learn and its implementations of several linear regression algorithms as part of our "predicted rating" advanced functionality
  - We chose scikit-learn because it is written in python and works well with numpy
  - scikit-learn was fairly straightforward to use with decent documentation available online

5. Basic Functionalities

SEARCH MOVIES

- A user types in a movie title and we ask our database to search through it and return all results that contain the search string they put in
- Note that this may take a while since the string they put in could be a substring of a movie so the whole table would have to be searched
- This is important because the whole point is to search for a movie and see what people think about it

SELECT title, genres, id FROM movie2 where $search_category like '%$search_term%'


MOOVIES

the hobbit

(● Movies) ○ People

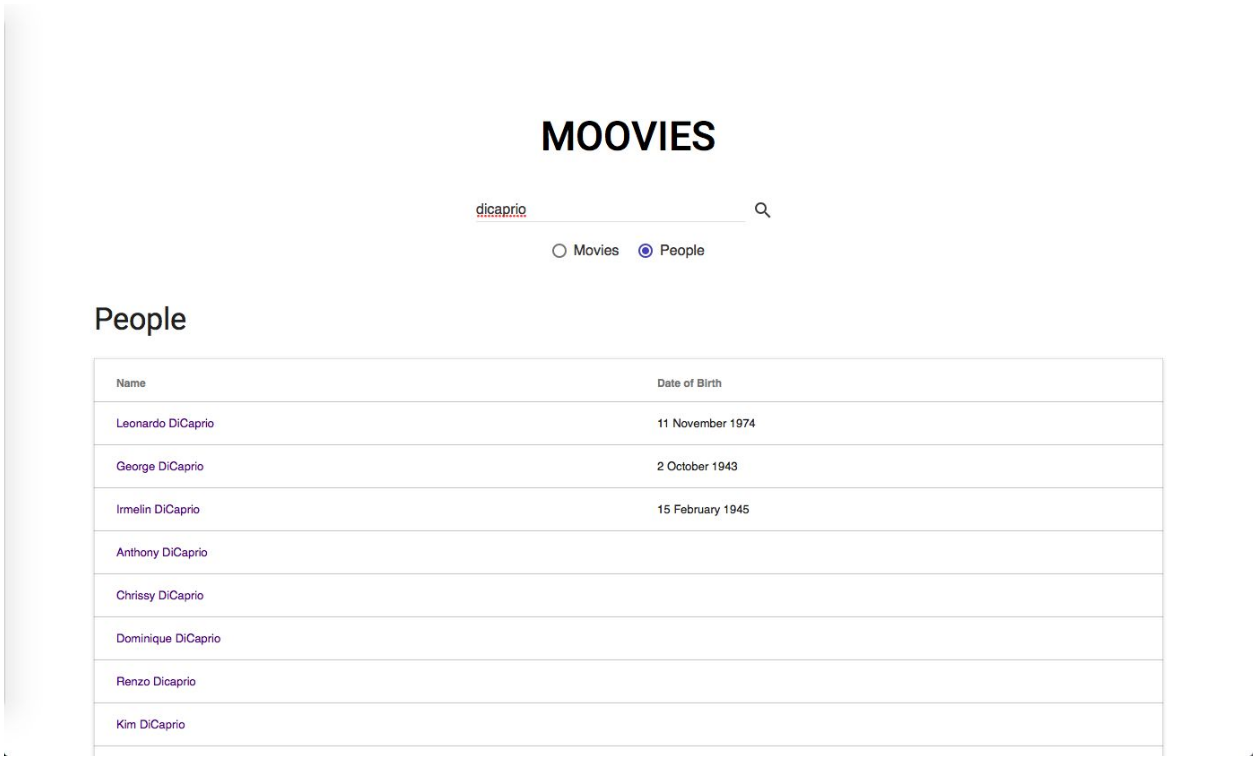
Movies

Title	Genre
The Hobbit	Animation, Short
The Hobbit an Unexpected Job	Action, Adventure, Short
The Hobbit Games	Comedy, Short
The Hobbit: An Unexpected Journey	Adventure, Fantasy
The Hobbit: Kingdoms of Middle-earth - Dance Battle	Adventure, Short
The Hobbit: The Battle of the Five Armies	Adventure, Fantasy
The Hobbit: The Desolation of Smaug	Adventure, Fantasy
The Hobbit: The Swedolation of Smaug	Adventure, Fantasy, Mystery

SEARCH PERSON

- A user types in a person they want to look for, they can search for first name or last name or part of a name and the relevant results will be displayed ordered by how many movies a person has worked on (acted in/directed)
- Note that this search also looks through the entire table because we are, once again, returning the results of a search on a possible substring
- This is important because if a user likes a movie because of a particular actor/actress they may want to look at other movies that person was in

SELECT first_name, last_name, id, dob FROM person2 WHERE ((person2.first_name like '%$name_split[0]%' and person2.last_name like '%$name_split[1]%') or (person2.first_name like '%$name_split[1]%' and person2.last_name like '%$name_split[0]%')) order by num_movie_associations DESC



People

Name	Date of Birth
Leonardo DiCaprio	11 November 1974
George DiCaprio	2 October 1943
Irmelin DiCaprio	15 February 1945
Anthony DiCaprio	
Chrissy DiCaprio	
Dominique DiCaprio	
Renzo Dicaprio	
Kim DiCaprio	

More Functionalities (not considered basic)

UPDATE

- This is used to update our the retweet count in our tweets table since some people may have retweeted since the last time we got the data
- This is important so we can get the newest up to date information and display the popular tweets

UPDATE tweets SET retweet_count = '$retweet_count', WHERE tweet_id = '$tweet_id' AND retweet_count <> '$retweet_count'"

INSERT

- We want to be able to add new tweets to our database so every time twitter gives us some new tweets, we want to see where the tweet is coming from and update our map to show the number of tweets per state about the particular movie

INSERT INTO tweets (movie_id, tweet_id, text, location, fips, timestamp, user, screen_name, retweet_count) VALUES ('$id', '$tweet_id', '$text', '$location', '$fips', '$timestamp', '$user', '$screen_name', '$retweet_count')"

SEARCH and JOIN

- When a user searches for a person, we want to provide them with a list of all movies they are associated with (acted in/directed) so in order to do that we would need to provide a join between the movie table and the worked on table by using the person_id we already got from the user searching for a person
- This is important because the user may be interested in movies that a particular director/actor/actress was in

*SELECT movie2.title,movie2.id FROM movie2 INNER JOIN (SELECT * FROM worked_on WHERE person_id=$id GROUP BY movie_id) as A on movie2.id=A.movie_id ORDER BY movie2.predicted_rating DESC"*

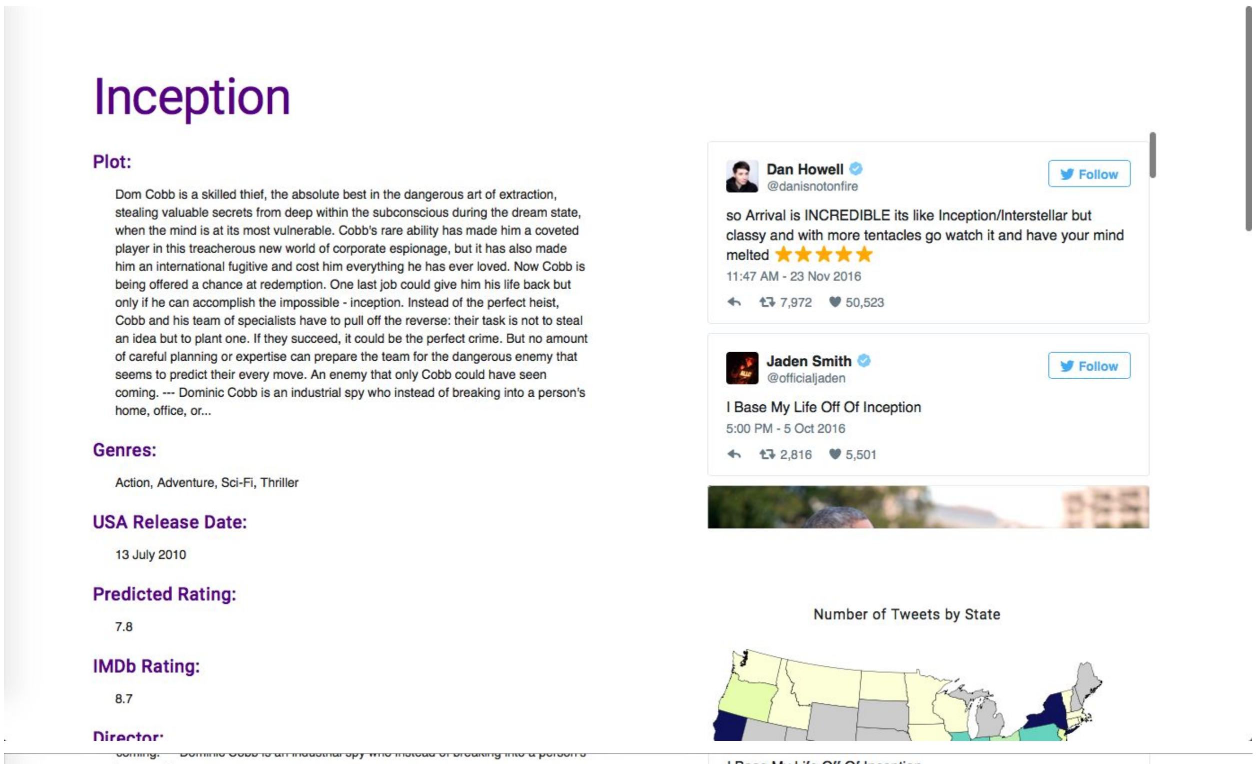
6. Advanced Functionalities

Advanced Functionality One: Twitter:

We believe that this is important for our application because it provides visual feedback for the user. This also gives them a somewhat different type of information. Instead of reading a bunch of words, they can look at a picture and see right away how popular a movie may be and who is talking about it and where it is popular and why it is popular. This also makes conveying information to the user more concise and simpler.

This functionality is slightly more interactive because the user can scroll through top tweets and see pictures of what people posted when they talk about a movie and then they can form their own opinion about it based on the visuals

Additionally the map will tell the user where a movie may be more popular in so they can decide if they should see the movie/check it out in order to have something to talk about with other people in their area, it'll keep them up on current social news and public opinion



Inception

Plot:

Dom Cobb is a skilled thief, the absolute best in the dangerous art of extraction, stealing valuable secrets from deep within the subconscious during the dream state, when the mind is at its most vulnerable. Cobb's rare ability has made him a coveted player in this treacherous new world of corporate espionage, but it has also made him an international fugitive and cost him everything he has ever loved. Now Cobb is being offered a chance at redemption. One last job could give him his life back but only if he can accomplish the impossible - inception. Instead of the perfect heist, Cobb and his team of specialists have to pull off the reverse: their task is not to steal an idea but to plant one. If they succeed, it could be the perfect crime. But no amount of careful planning or expertise can prepare the team for the dangerous enemy that seems to predict their every move. An enemy that only Cobb could have seen coming. --- Dominic Cobb is an industrial spy who instead of breaking into a person's home, office, or...

Genres:

Action, Adventure, Sci-Fi, Thriller

USA Release Date:

13 July 2010

Predicted Rating:

7.8

IMDb Rating:

8.7

Director:

Dan Howell @danisnotonfire Follow

so Arrival is INCREDIBLE its like Inception/Interstellar but classy and with more tentacles go watch it and have your mind melted ★★★★★

11:47 AM - 23 Nov 2016

7,972 50,523

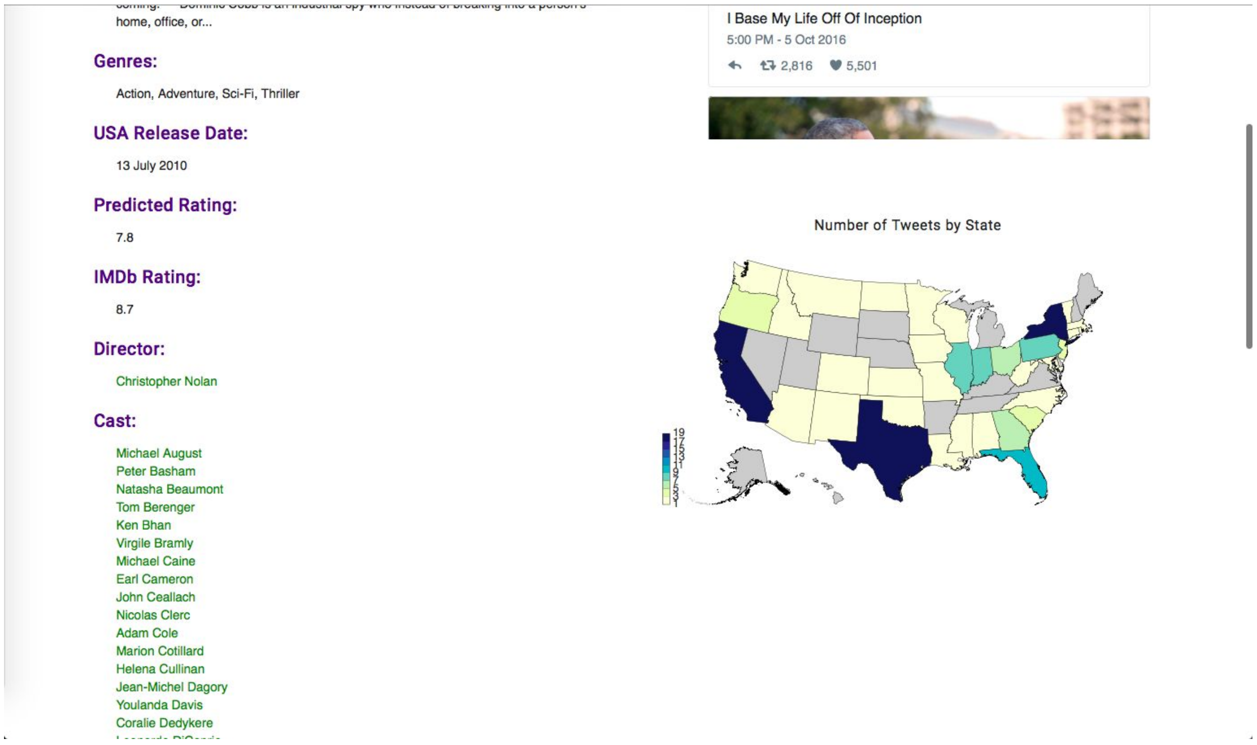
Jaden Smith @officialjaden Follow

I Base My Life Off Of Inception

5:00 PM - 5 Oct 2016

2,816 5,501

Number of Tweets by State



home, office, or...

Genres:

Action, Adventure, Sci-Fi, Thriller

USA Release Date:

13 July 2010

Predicted Rating:

7.8

IMDb Rating:

8.7

Director:

Christopher Nolan

Cast:

Michael August
Peter Basham
Natasha Beaumont
Tom Berenger
Ken Bhan
Virgile Bramly
Michael Caine
Earl Cameron
John Ceallach
Nicolas Clerc
Adam Cole
Marion Cotillard
Helena Cullinan
Jean-Michel Dagory
Youlanda Davis
Coralie Dedykere

I Base My Life Off Of Inception

5:00 PM - 5 Oct 2016

2,816 5,501

Number of Tweets by State

Algorithm/Solution

1. Get API key and authorization: figure out how to use it/what is it necessary for
2. Learn how the API documentation works and then use it to get the twitter data
3. Parse the information and put the data into our database table called tweets

4. Run queries to get the retweet count and display the top ten (we had to figure out the top 10 by ourself)
5. Learn how to use twitter's embedded timeline so we can display the tweets
6. Parse location data from twitter and store it into our map data structure to use for our map of tweets in the U.S.
7. Figure out how to open csv file and permissions in order to save data for our map
8. Figure out how to use D3's documentation and functions in order to create our map and display it

Evaluation

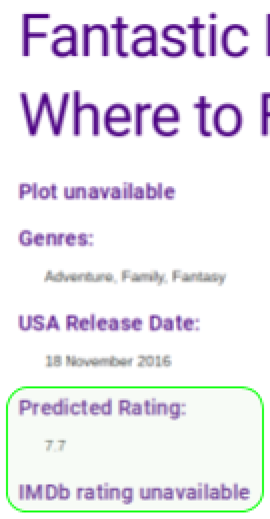
- Goal: The reason we worked with Twitter was to make it more visual and provide more relevant information to the user. The top 10 tweets allow the user to see what other people think of the movie and the map shows how many people talk about/know the movie in the state they are from (if they are from the U.S.)
- This sets us apart from all the other applications because they are not as visual with the information they are trying to provide. People are visual creatures and having a picture or graph makes it seem more interesting and eye catching.
- Additionally, this makes our application seem more interactive and tailored to the user. For example, IMDb gives a lot of information about a movie but the information is mostly in words. Additionally, we are slightly more interactive because of the tweets and the map.
- They can develop a better opinion about the movie by looking at the twitter data instead of having to search other sites that show what other people think.

Challenges

1. Twitter wasn't very developer friendly so we had to not only use their Documentation but by a lot of trial and error, parse some of the data ourselves and figure out what information was useful
2. Using Twitter's embedded timeline to display a tweet was difficult because the Documentation wasn't clear and we wanted it in a specific format
3. We also needed to decide which tweets were most relevant to the user's search so we decided that a tweet had more priority/was more relevant by its retweet count
4. While getting the necessary data for our map was easy, trying to make a csv file to load relevant information for our map took a really long time because of file permissions

Advanced Functionality Two: Prediction Algorithm

IMDb uses user ratings to provide an aggregate score for each movie ranging from 0-10. Unfortunately, not every movie has enough user ratings to provide a score, either because the movie was not popular or has not yet been released to the public. Our prediction algorithm attempts to solve this problem by enabling the MOOVIES application to predict the score for each movie in our database. The predicted rating for each movie is listed next to the IMDb rating - if available - on our website.



Fantastic Beasts and Where to Find Them

Plot unavailable

Genres:

Adventure, Family, Fantasy

USA Release Date:

18 November 2016

Predicted Rating:

7.7

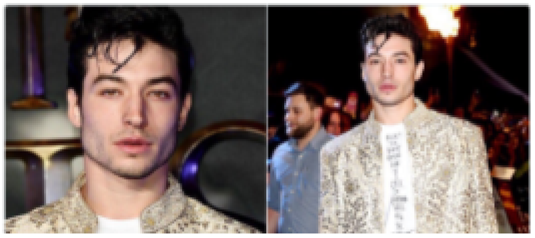
IMDb rating unavailable

Shawn Mendes @ShawnMendes

Just watched Fantastic Beasts and Where To Find Them and it completely sparked my love for HP. Time to marathon. Haha

8:40 PM - 25 Nov 2016

27,325 82,017



Not enough users have rated this movie, but our prediction algorithm provides an estimated score of 7.7

Our prediction algorithm uses machine learning techniques, specifically linear regression algorithms, to label each movie with a predicted rating. We use an SQL query with several joins to select attributes for each movie that may be relevant to its rating. In particular, for each film we combine its genres, cast/director, and plot into a bit vector instance. An element of the vector is "switched on" (set to 1) if the given attribute is present in this particular movie, and set to 0 otherwise. For example, the movie *The Fast and the Furious* would have a 1 in its instance vector at the index corresponding to the actor Vin Diesel. During training, the label for an instance is taken from the IMDb rating for the movie (i.e. 6.7 for *The Fast and the Furious*). At a high level, the process for predicting a rating for each movie in the database is simple:

1. We construct instances and labels for each movie in the database that has an IMDb rating (~22,000 movies).
2. We learn a linear model representing the training data via a regression algorithm.
3. We evaluate the performance of the model on the training data, specifically calculating the average absolute error between predicted ratings and true ratings.
4. If we are satisfied with the accuracy of our model on the training set, we construct unlabeled instances for each movie in the database (~1 million movies).
5. We use the learned model to predict a rating for each movie, then commit these predictions to the database via SQL update queries.

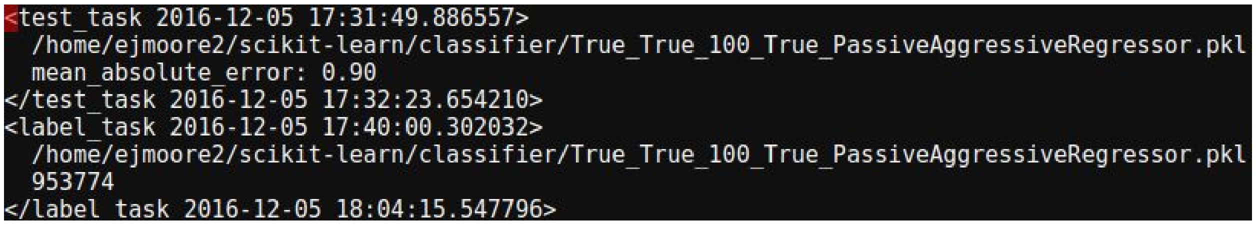
We believe our prediction functionality is quite unique. A quick internet search reveals that some others have attempted to predict IMDb movie ratings, but we are not aware of any major website that provides predicted movie ratings. Sites like Netflix suggest films based in part on a user's profile, but our algorithm operates independently of the user and tries to generate a score based solely on characteristics of the movie itself. In evaluating the performance of our model we used performance on the training set as our primary metric. Our best model yielded an average absolute error of 0.90 on the training set. This accuracy is not terribly impressive, but we think it is decent enough that our predicted ratings may at least provide some indication as to whether or not a movie is likely to be any good. We think that we may be able to improve the performance of our model with additional tuning of parameters and improving the generation of our feature space. Additionally, we suspect that a linear model may not be enough to capture the assumed underlying function determining movie ratings. For example, our model cannot handle the exclusive-or (XOR) case. Suppose that actor A and actor B have appeared separately in many high rated movies, but they have terrible chemistry and any movies they appear in together are very poor. Our linear models lack the expressivity to predict poor scores in the case that A and B work together, but good ones when they work alone.

We faced numerous challenges in implementing rating prediction for our MOOVIES application, beginning with machine learning. We initially attempted to use custom implementations of neural networks, linear regression, and perceptron. For perceptron we used 5 as a threshold rating, classifying movies below this rank as “poor” and those above as “good”. However, we eventually moved to scikit-learn for implementations of various regression algorithms. This allowed us to experiment with more learning algorithms and parameter settings. The main challenge with rating prediction was the scale of data we were using, and the implications this had on both space and time complexity. As an example, our final instance generation algorithm creates numpy arrays with length of over ~200,000 for each movie. Our initial attempts to generate instances failed due to “out of memory” errors, or inordinately long runtimes. As such we implemented several strategies for reducing memory usage and reducing runtime for our algorithm:

- In constructing the feature space for the cast of a movie we only considered people who appeared in at least *k* movies, since successful actors would have a bigger impact on a film’s success. This significantly reduced the size of our feature space for higher values of *k*.
- In constructing the feature space for plot we only considered words that were at least *k* characters long, and we normalized by taking the lowercase of each word.
- In each case, constructing features for genres, cast/director, and plot required building a consistent dictionary mapping the movie attribute to an index in our bit vector. To avoid duplicating work during the training, testing, and labeling processes we saved each dictionary to a file after initial construction. This allowed us to avoid repeating expensive SQL queries involving multiple joins, but rather simply read from files instead.
- In general, we used buffered queries on our database when constructing a set of instances for training, testing, or labeling. In labeling for example, we needed to generate and label almost a million instances, so we would construct instances in batches of a much smaller, predetermined size so as to limit our memory footprint.
- When labeling, we would use our model to predict the ratings for a batch and add the predictions and movie IDs to a growing list of all the new predictions. When the list grew past a certain size (~10,000 entries) we would append all the ID, rating pairs to a

temporary file and clear the in-memory list. At the end of the labeling process we issued SQL update queries for each pair in the memory buffer, then read from the file and executed update queries for all the saved pairs. Then we committed all updates to the database at once and closed our connection.

Even with all our optimizations, processes still took a significant amount of time. Training and testing for example required around 10 minutes, and labeling could take several hours. In addition, we wanted to be able to re-run the prediction algorithm regularly if new rated or upcoming movies were added to the database. Therefore, we constructed separate scripts for training, testing, and labeling with a given set of parameters from a configuration file. Then we used the Linux crontable to regularly execute the scripts, dumping their output to a log file. Reviewing the log file gave use information about when a task had begun, what parameters it was using, what its effect was and if it completed. An example from the log is below:



```
<test_task 2016-12-05 17:31:49.886557>
  /home/ejmoore2/scikit-learn/classifier/True_True_100_True_PassiveAggressiveRegressor.pkl
  mean_absolute_error: 0.90
</test_task 2016-12-05 17:32:23.654210>
<label_task 2016-12-05 17:40:00.302032>
  /home/ejmoore2/scikit-learn/classifier/True_True_100_True_PassiveAggressiveRegressor.pkl
  953774
</label_task 2016-12-05 18:04:15.547796>
```

Two entries from "cron.log", showing the results of a test task and a label task

7. Future Work

- One possible improvement would be to add pictures to people and move pages. Our database contains a lot of movies and people with similar titles/names and it can sometimes be hard to find exactly what you're looking for without a visual guide.
- Another improvement would be to make our map more interactive by allowing a user to click on a state and having tweets from that state show up so they can get more information on what people from certain states think about a movie

8. Work Division

Nga Thai:

- Worked with Twitter API to display popular tweets and store it into our database
- Used D3 to display a map that shows the number of tweets from each state relating to a particular movie

Edward Moore

- Used a prediction algorithm to generate a predicted rating for each movie that compares to the IMDb rating

- Used Machine Learning to study our data and try to make it better/accurate

Max Li:

- Worked with Twitter API to gather the relevant information for each movie
- Developed movie and person pages and implemented some functionalities such as search and join

Omar Abdulaev:

- Designed our web application using Material Design Lite, worked with the CSS and HTML portion
- Helped implement the basic functionalities such as searching for a movie/person

Lessons Learned About Teamwork:

- Teamwork is nice because everyone helps each other
- If someone didn't know something, someone else might know/could help figure it out
- Learn new ideas and good brainstorming, change/improve our approach/ideas on implementations and methods
- Being flexible with time