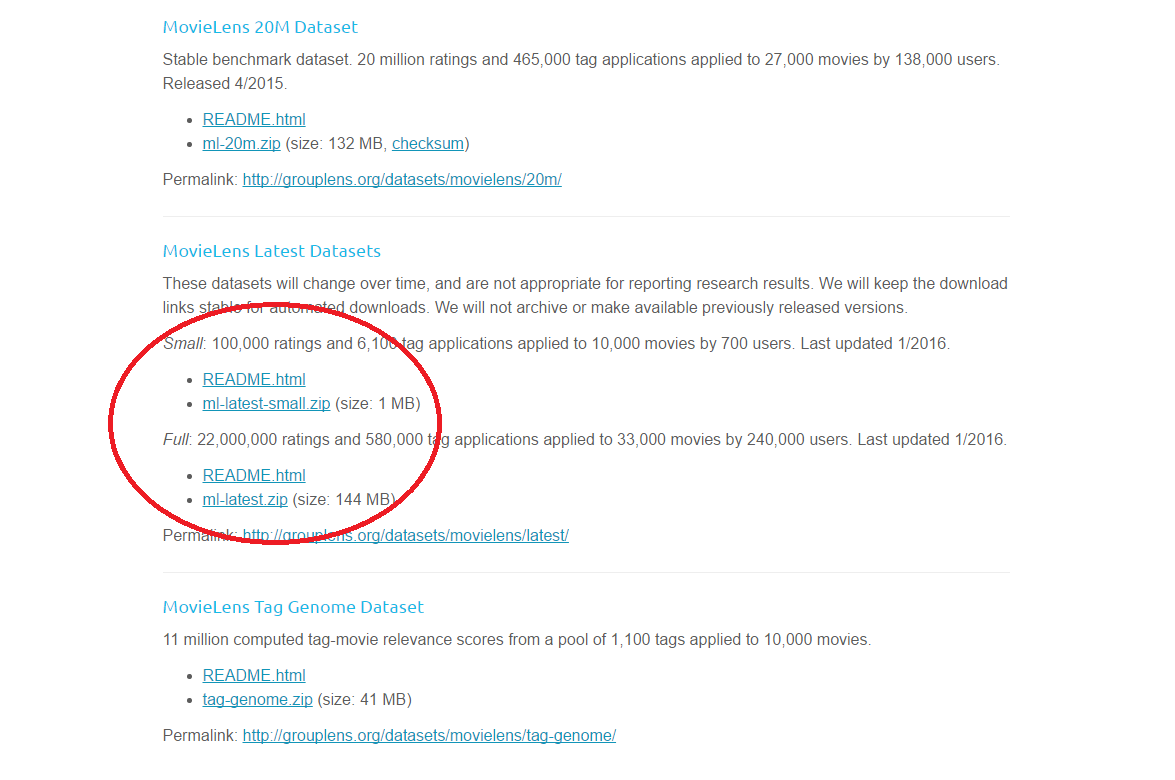
# User Customizable Movie Recommendations

## Project Purpose:

The goal of this project is to develop a robust movie recommender system in which users have active control over the recommendations they receive. Recommender systems are not a novel problem, being both prevalent in many applications and well researched, largely due to the Netflix Prize Competition. However, very few of these systems allow the users to fine tune the recommendations they receive. Giving a user more freedom and control (beyond just imposing a filter) should improve the user experience. This recommender system, like most other recommender systems, can generate a list of movies a user would likely rate highly and quantify how similar two movies are. What really sets it apart is its ability to let a user to choose how much emphasis they would like to apply to genres and release year when searching for movies similar to a target movie.

## About the Data:

MovieLens.org is a website that allows users to provide ratings and tags to movies. It is operated by GroupLens, a research group from the University of Minnesota. For this project, I chose to use the Full and Small versions of the Latest Dataset available at <http://grouplens.org/datasets/movielens/>, updated as of January 2016.

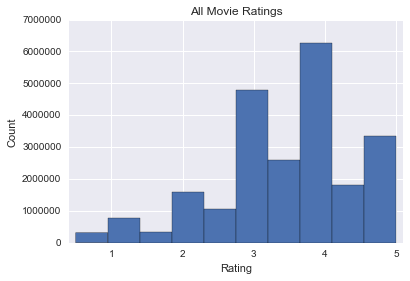


# Insights from Data Exploration

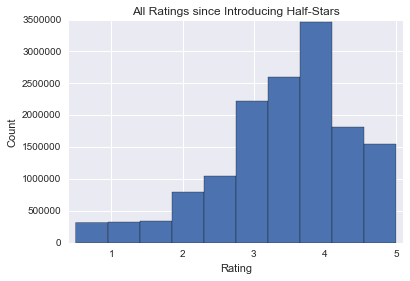
Prior to building the recommendation system, an exploratory survey of the Full data set revealed a few interesting and impactful insights which are highlighted below.

## Movie Ratings:

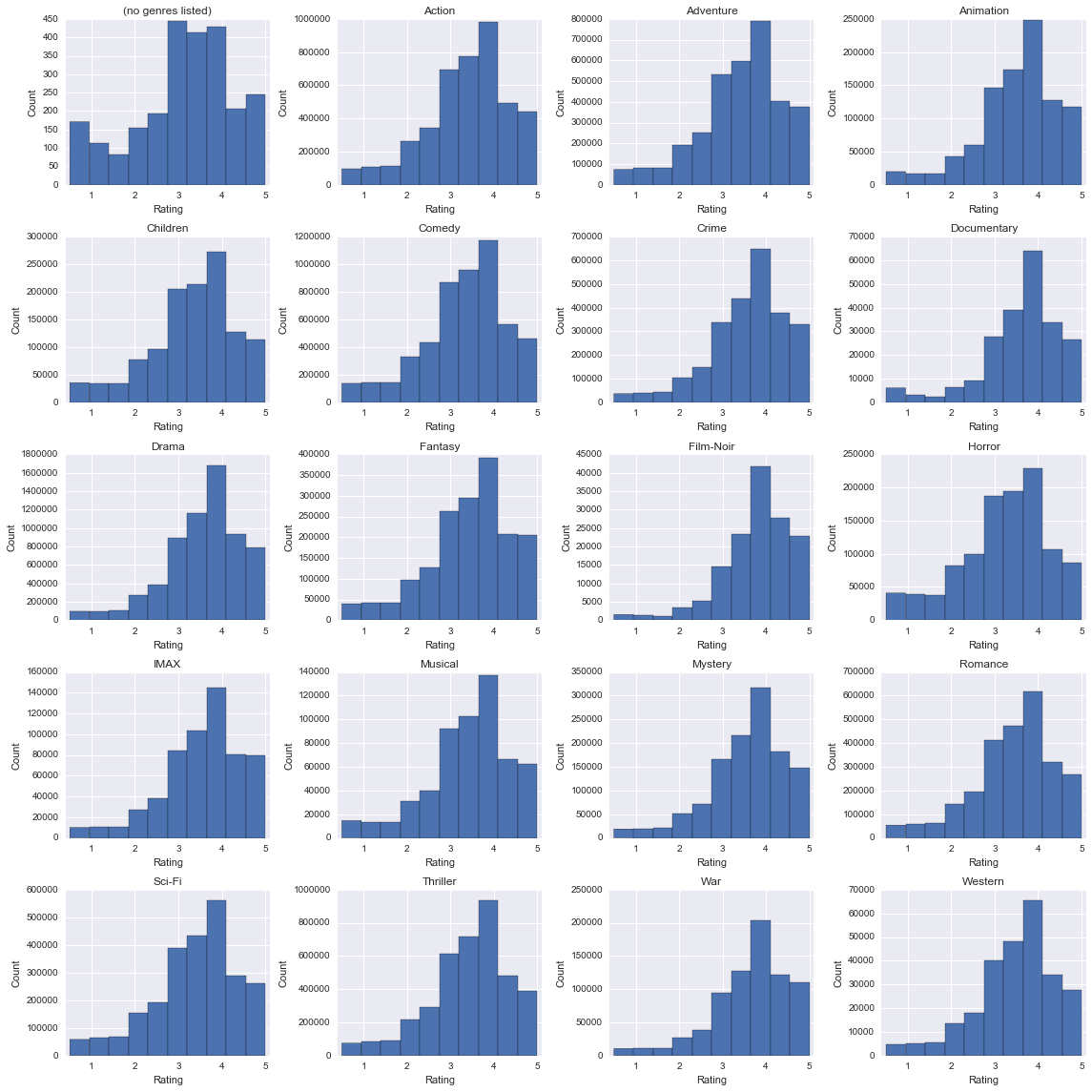
One of the first things that jumps out about this dataset is the odd distribution of the ratings across all movies:



The easily noticeable bias towards whole-number ratings does have an explanation: the half-star ratings were not implemented until a much later date. In fact, the earliest half-star rating occurs in February 2003, whereas the dataset’s earliest ratings start in January 1995. Because this has such a large effect on the distribution of ratings, when training the recommendation system, it will be necessary to correct for this bias. Therefore, prior to training any model, we must filters out data prior to August 2003, allowing for a six month adoption period. Correcting for this discrepancy, we arrive at the following distribution:



The result, a left skewed distribution, more closely matches what was expected. Considering that people will tend to watch something they already have an interest in, there should be an inherent bias towards higher ratings. Few people will intentionally take the time to watch a movie that they have no interest in and then rate it. This distribution remains prevalent across an examination of the movie genres as depicted in the histograms below:



## Regarding Genres:

Moviegoers often identify highly with specific genres that match their interests. As such, genres are very important to a user’s decision to watch a movie or not. The MovieLens dataset classifies all movies into a combination of the following genres:

|  |  |  |
| --- | --- | --- |
| '(no genres listed)' | 'Action' | 'Adventure' |
| 'Animation' | 'Children' | 'Comedy' |
| 'Crime' | 'Documentary' | 'Drama' |
| 'Fantasy' | 'Film-Noir' | 'Horror' |
| 'IMAX' | 'Musical' | 'Mystery' |
| 'Romance' | 'Sci-Fi' | 'Thriller' |
| 'War' | 'Western' |  |

Only movies without a genre will go in to the ‘(no genres listed)’ category. This isn’t a true genre as it doesn’t have anything to do with the contents of the movie. By that same logic, ‘IMAX’ isn’t truly a genre either as it has more to do with the equipment used to record and/or project the movie. Furthermore, not all people watching a movie will watch it in IMAX. Therefore, both of these ‘genres’ are ignored in further exploratory work.

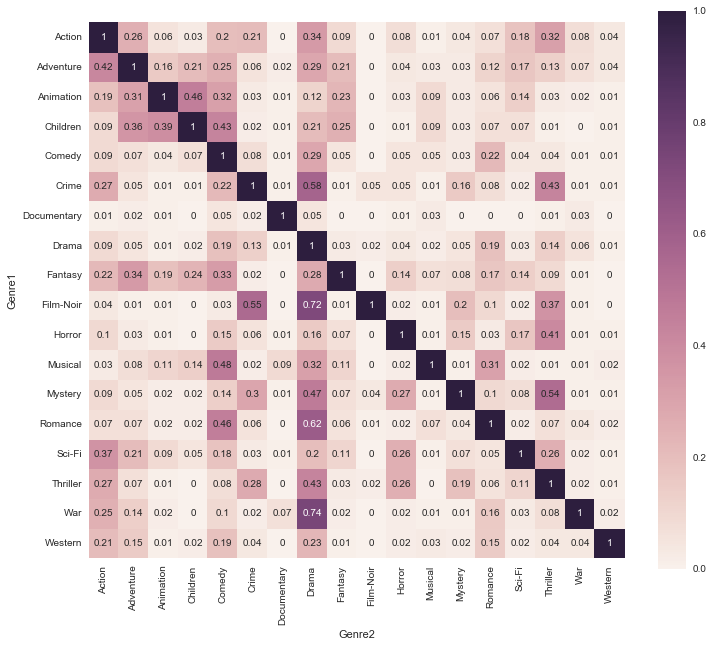
Genres pose an interesting challenge because they are essentially classifying the movies into different categories with one exception: movies can belong to multiple categories at once. This goes against the very definition of a categorical variable. Instead of using them as a categorical variables, it is possible to convert this information into a ‘genre similarity’ score by measuring the overlap between two specific genres as seen in the figure below. It essentially determines what percentage of movies that are identified as belonging to Genre1 also belong to Genre2.

A quick glance at the Figure below allows us to draw some quick conclusions:

1. Some genres have very little overlap with others. In particular, documentaries stand well apart from the other genres. This is to be expected considering that the tone and purpose behind a documentary is generally to educate and inform the audience rather than to entertain with a story.
2. Most genres have a sizeable overlap with Drama (particularly Crime, Film-Noir, Romance, and War), but the reverse is not true. The drama genre, as its defined, contains stories about realistic characters facing realistic struggles, a rather general theme which appears across most movies.
3. There are sizeable overlaps in some of the expected genre pairings, for example:
   * Adventure and Action
   * War and Drama
   * Romance and Comedy
   * Mystery and Thriller

This only really serves to indicate that some genres function well together, while other pairings have either not been thoroughly explored or are just not meant to be combined (Children and War).

While these genre overlaps are interesting to study on their own, they could also be used to enhance the recommender system by guiding it to movies with a higher genre similarity.



# Building a Recommender System

## Data Filters:

Prior to building the model, a review of the raw data shows that some of the data should not be used when training and testing the recommender system:

1. As mentioned above, the raw data has a bias towards whole-star ratings due to the late implementation of the half-star option. All ratings prior to August 2003 are filtered out to correct the discrepancy and allow for a 6 month adoption period.
2. There are a few users who seem to have rated upwards of 2000 movies making them significant outliers by nearly doubling, or in a few instances, over doubling the number of ratings when compared to the next most active user. Due to the rather suspicious amount of activity from these users, they were filtered out of the dataset.
3. In order to properly train, validate, and test the data, all partitions must contain many ratings per movie and per user. The model uses 80% of the data for training and 4-fold cross-validation, while the remaining 20% is held back for testing. In an effort to assure that each partition contains at least five ratings per movie and per user, users with fewer than 40 ratings were filtered out. Then, movies with fewer than 35 ratings were also removed. The second filter is applied on the results of the first meaning that some users may have lost some of their ratings, hence the higher threshold (40 versus 35). These threshold values were picked in an effort to ensure that all partitions had over five ratings per movie and per user (20% of 35 is 7, allowing for some margin of error given the random nature of splitting the data). Each 20% of the data should, in theory, should have at least 7 entries per movie and user, but random partitioning may make it less which is why there is a small margin for error in this step. These were the lowest sensible threshold values possible, because increasing the thresholds, while likely better for accuracy, would filter out too much of the viable data available.

The combination of all these adjustments to the raw data have significantly lowered the size of the Small dataset from 105,339 to 20,816 ratings.

## Performance Metrics:

Similar to the Netflix Prize Competition, this project uses Root-Mean-Square-Error as a metric to evaluate model performance. The measure of success is the extent to which the recommender system can reduce the RMSE in comparison to the average movie rating (trivial model), the simplest reasonable recommendation possible. Despite controversy over RMSE as a performance metric, this project assumes a claim that “a 1% improvement of the RMSE can make a big positive difference in the identity of the "top-10" most recommended movies for a user” made on the [Netflix Prize Forums](http://www.netflixprize.com/community/viewtopic.php?id=828) is true, and therefore a valid metric for analyzing model performance.

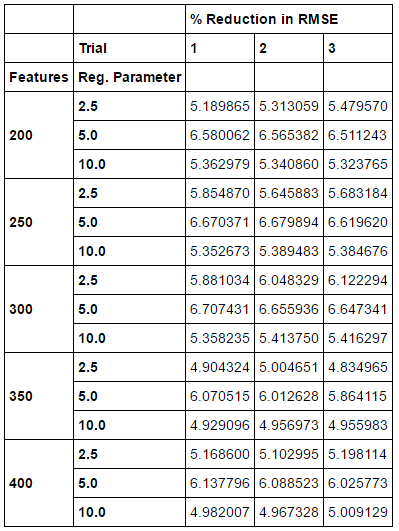
For Netflix Prize, the goal was to improve upon (reduce) the RMSE of Netflix’s Cinematch algorithm by over 10%. The data resulted in an RMSE of 1.0540 on the training data when providing just the average rating. The Cinematch algorithm achieved an RMSE of 0.9514 on the quiz set (accessible to participants), and 0.9525 on the test set (unavailable to participants). Therefore, the target RMSE for participants in the competition was 0.8563 on the quiz data and 0.8572 on the test data, approximately 18.7% below the trivial model of just providing the average. Two teams managed to achieve this goal within 3 years of the competition’s debut.

Considering the MovieLens Small data is completely different and was prepared differently from the Netflix Prize data, the best way to draw a comparison between the work presented here and the work in the Netflix Prize competition is by comparing the reduction in RMSE. The RMSE for the trivial model on the training partition is 0.8535, already considerably lower than the Netflix Prize’s training data. As discussed below, the final model will lower this RMSE by 7.85% on average to approximately 0.7865.

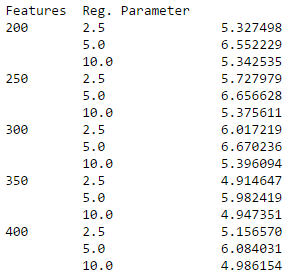
## Optimizing and Training the Recommender System:

The model is being trained from a completely random initialization of both movie and user features. An estimate of what a single user will rate a single movie can be obtained by the sum of products between that user’s and that movie’s features. This permits for the use of gradient descent in a collaborative filtering algorithm, adjusting both the user and movie features every iteration.

There are two key parameters that needed to be optimized, the number of features (per user and movie) and the regularization parameter. As mentioned above, 4-fold cross-validation was used on an 80% partition across a variety of parameter values. The 4-fold cross-validation essentially sets a quarter of the data aside for scoring and trains the model on the remaining three quarters. After storing the results (reduction in RMSE compared to providing the average rating), it sets aside the second quarter of the data and trains a new version of the model on the remaining data. This is repeated four times so that each quarter of the 80% partition acts as the cross-validation data once. The results are then averaged to determine the model’s performance. This 4-fold validation was run for every pairing of regularization parameter (2.5, 5, or 10) and number of features (200, 250, 300, 350, 400). This entire process was repeated two more times so that each pairing was 4-fold cross-validated in three separate trials. The three trials are a precautionary measure due to the fact that every model trained begins from a fresh random initialization and that all of the solutions the model determined are likely to be local optima. A summary of the results is as follows:



By averaging across the three trials per parameter pairing and selecting the highest % Reduction in RMSE, we get the optimal parameter values of 300 features and 5 as a regularization parameter:



These parameters were then used to train five more models using the full 80% training partition. These five models were scored against the 20% testing partition and their results averaged to report a 7.85% reduction in RMSE when compared to the trivial model of providing a user with the movie’s average rating.

The Full dataset, due to processing limitations could not use the same batch gradient descent algorithm as the Small dataset. Although a separate mini-batch gradient descent algorithm has also been developed, there are no results to report as it could take weeks to fully optimize and train the model on such a large dataset.

## Making a Personalized Movie Recommendation

If a user takes the time to rate several movies, the model is now capable of providing a personalized list of recommendations by predicting that user’s rating across all movies included in the data. This is achieved by performing linear regression to learn the new user’s features based on the movie ratings they have provided. The linear regression does require its regularization parameter to be optimized. For this optimization, there is not really any need for a cross-validation or testing partition. The user features will always converge towards a global minima that varies solely with the regularization parameter. Having learned the user’s features, predictions of that user’s ratings are readily available by taking the dot product of the each movie’s features and the new user’s features. The most time intensive part of this process is getting the user to rate their watched movies, but once that is done, the recommendation system can quickly generate a list of personalized movie recommendations:



Similarly, users who were filtered out because they were below the threshold of 35 ratings can be provided with ratings using linear regression and the learned movie features. The more ratings they have, the better the accuracy. Because mean normalization was used in training the model, the predicted rating for a new user with no ratings would be the average rating across the training data for each of the movies. The model can completely work around the user threshold and all users can get a list of recommendations.

Movies that were filtered out because they failed to reach the threshold could also be reintroduced to the system by learning their features from the trained user features using linear regression. However, in contrast with the users that were filtered out, there should still be a minimum threshold on movies. Rating predictions generated from a basis of just a few observations are not as trustworthy and should not be passed on to the users. For movies that failed to reach the threshold, it would be better to give users the same rating as the trivial model, the movie’s average rating.

## Generating a List of Similar Movies

Given one movie, the algorithm is capable of generating a list of similar movies. To enable this recommendation, the model must calculate ‘distances’ between every possible movie pairing by using the respective movie’s learned features. Because the features may be highly correlated, it is best to start the process by performing Principal Component Analysis (PCA) and retaining 95% of the data’s variance. Apart from compressing the features and making them easier to visualize, PCA also maps the correlated movie features onto orthogonal co-ordinates making them linearly uncorrelated. Now, the newly mapped features can be used to calculate a ‘distance’ between two movies. The smaller the distance, the more similar the movies.

However, before calculating the distance, there is one other concern to address: the varying scale of the principal components. This is easily addressed by normalizing the values on a scale of 0 to 1. Finally, we can determine the Euclidean distance between the normalized principal components to assign a numeric measure to similarity between movies. One last and optional step, scaling (not normalizing) the distances down to a range of 0 to 1, makes future work easier to conceptualize. After employing all of these calculations, the model is able to quickly generate a list of movies similar to said target. As a running example, all work shall be demonstrated on the movie “Harry Potter and the Philosopher’s Stone” (HP1), released in 2001 and listed as an Adventure, Children, and Fantasy film:

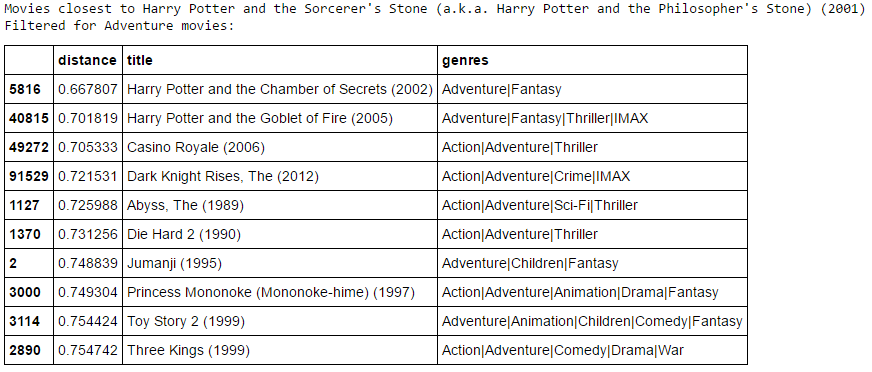


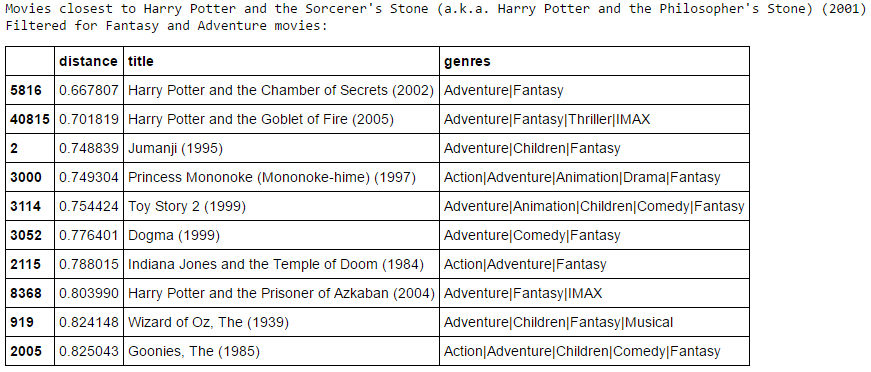
## Post-Processing and User Customization of Similar Movies

As it stands, the list of similar movies could use some refinement. Just looking at the list, there are some questionable entries from a human’s perspective. One way to resolve this is by giving the user the ability to fine tune their recommendations.

### Exclusive Filters

For example, allowing the user to filter for movies belonging to one or more genre can yield:

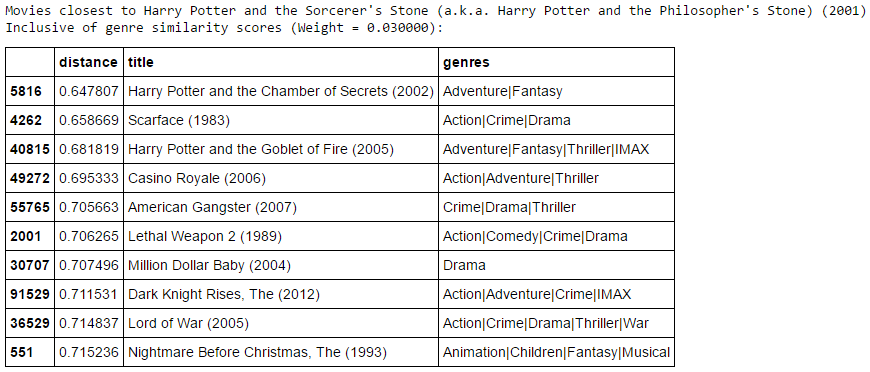




While some users may appreciate the ability to filter out results based on genres, it is an extremely steep penalty to apply to movies without those specific genres. There may be truly similar movies that get completely filtered out because they don’t list a specific genre. For example, if a filter for Children’s films were applied, the remaining Harry Potter films would be filtered out too in the example above.

### Genre Similarity Score

Alternatively, the recommendations could be enhanced by generating a genre similarity score to compliment the distances between movie features. To determine a genre similarity score, we determine what percentage of the target movie’s genres exist in each of the other movies. This similarity score is then multiplied by a weight constant (user choice) and then subtracted from the distance. This constant has a lot of control over the resulting recommendations:







### Genre Overlap Score

The genre similarity score that was just discussed can still be considered rather rigid, disproportionately decreasing the ‘distance’ for some movies while movies without a common genre don’t move at all. In the exploratory analysis discussed above, we were able to see how much of an overlap exists between one genre and the others across all movies in the MovieLens’ Full dataset. The genre overlap score is inspired by that analysis. It assumes a relationship between genres, quantified by their overlap. Thus, we can derive a genre overlap score by averaging the overlap across all the genres in the target movie with all the genres in a second movie. For example, HP1 (Adventure, Children, Fantasy) and the movie “Scarface” (Action, Crime Drama) had a genre similarity score of 0, but gets a genre overlap score of 0.18 (see below). Meanwhile, HP1 and “Jumanji” (exact same genres as HP1) had a genre similarity score of 1, but a genre overlap score of 0.51:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Scarface** | | | **Jumanji** | | |
| **HP1** | Action | Crime | Drama | Adventure | Children | Fantasy |
| Adventure | 0.42 | 0.06 | 0.29 | 1 | 0.21 | 0.21 |
| Children | 0.09 | 0.02 | 0.21 | 0.36 | 1 | 0.25 |
| Fantasy | 0.22 | 0.02 | 0.28 | 0.34 | 0.24 | 1 |
| Average | 0.18 | | | 0.51 | | |

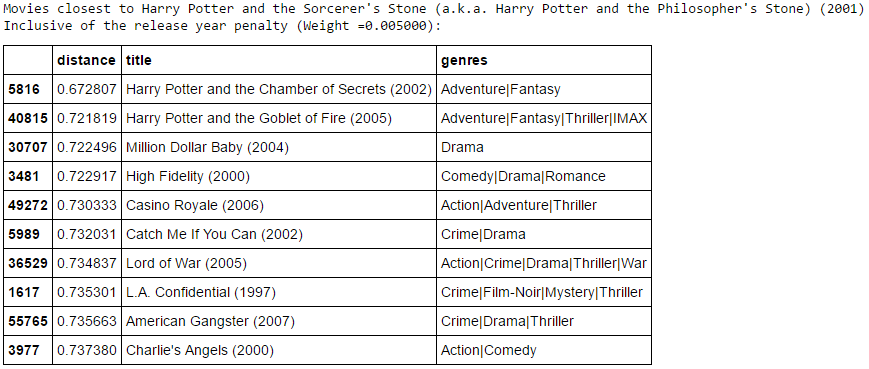
These values are, again, multiplied by a weight constant (user choice) and then deducted from the distance scores:



Similar to the genre similarity score, the genre overlap score’s weight constant has a significant amount of control over the results. In both cases, the weight constant, if not capped, could be used to overshadow the initial distances calculated from the trained movie features.

### Release Year Penalty

Another tweak, penalizing the absolute difference in the release year of two movies, could be useful for users who want to emphasize similar movies from around the year the target movie was released. Again, it’s possible to put the control in the user’s hands through a weight constant. This weight constant needs to be much smaller because the difference in release years operates on a much larger scale.



### Combining Customization Options

The best option to maximize a user’s control would be to implement a combination of a genre-based score and a release year penalty. With separate weight controls, the user can truly emphasize the aspects of the recommendation they are most interested in. If the user has no interest in customizing their results, a weight of 0 would return the results to the default distances determined from the model’s parameters.



# Ideas for Improvement

As with all learning algorithms, this recommender system is limited by its inputs. In this case, the data used to actually build the model only spans 337 movies and 255 users, a tiny fraction of the available data from the Full dataset. Though I have built functioning stochastic and mini-batch gradient descent algorithms that could tackle the Full dataset, they both require a lot more time to optimize and debug in comparison to the current implementation. Due to the time and computational constraints, the Small dataset had to suffice for now.

# Precautionary Note

A completely random initialization, as this model uses, can generate slightly different results each time it is run. The features that the model trains could have their order randomized each iteration, or they might turn out to be an entirely different set of features. These fluctuations trickle down into the analysis for determine the ‘distance’ between movies. As such, the list of similar movies can vary quite drastically from one iteration to the next. The personalized movie recommendations, formed from both movie and user features, are nowhere near as volatile.

While this issue could be fixed by implementing a pseudo-random initialization for all of the parameters, doing so would lock the model into a single local minimum. This is undesirable for both model optimization and accurately reporting on the model’s performance.