# Nextflix: Movie Recommendations using MovieLens Dataset

Cyrus Tabatabai-Yazdi, Akshay Shetty and Akshay Raman
UCLA
ctabatab@ucla.edu, akshashe@ucla.edu, akshay.raman@cs.ucla.edu

***Abstract*— Recommendation systems are everywhere. Whether you are shopping online, reading news, or browsing Netflix, behind-the-scenes a recommendation system is trying to predict your next move before you even know it. One of the most popular techniques used for recommendations is collaborative filtering. Collaborative filtering comes in two flavors: memory-based and model-based. Memory-based is the traditional approach to collaborative filtering where past behavior is used to find similar users and items to make recommendations. Model-based approaches, made famous by the Netflix prize of 2006, depend on creating a model through machine learning or data mining algorithms for making recommendations. Both approaches have their benefits and drawbacks, and in this project, we examined both types as well as content-based recommendations. With the amount of data available on the Internet, finding a good recommendation is like finding a needle in a haystack, and our goal is to understand the different approaches used and their effectiveness by creating a movie recommendation system using the MovieLens dataset.**

## I. INTRODUCTION

Long gone are the days where customers are limited to buying whatever is available in a store. With the help of the Internet, the number of potential items a customer can buy nowadays is infinite. Next time you do an Amazon search for an item you want, notice the vast amount of results that turn up. How can an online shopper filter out the massive amount of information on the web to find a useful item or even discover something new? This information overload necessitates a process to filter the information for the hidden gems. Although search engines do provide some of this, search engines today are mainly focused on matching results to keywords rather than personalizing to what users might find interesting or useful. Thus, many online shops and services have spent years devising algorithms that can predict a person's taste and preferences for making recommendations of what to buy or do next. For many of these online businesses, their end game is to convert a recommendation to a sale or retention. Thus, it should come to no surprise that many interactions one might do online might be related to a recommendation. When you are browsing the news, the next article you end up reading could have been suggested. The next video shown automatically after a video finishes on YouTube is most likely a recommendation. From the time when Amazon used recommender systems to filter books from its vast catalog to today where it is the norm to have suggestions on what to read, watch or buy next, recommendation systems have significantly evolved and the rest of this report will demystify the magic behind these systems by building a movie recommendation engine using the MovieLens dataset.

## II. BRIEF HISTORY OF RECOMMENDER SYSTEMS

Before diving into the technical details, it is worth going over the history of recommender systems as it provides a foundation for understanding the reasoning behind systems today. One of the earliest recommender systems was the UseNet platform by GroupLens which recommended news stories to users based on what other users were reading [6]. GroupLens also created MovieLens in 1997, a platform where movie lovers could discover new movies based on the collaborative ratings of all users in the system. The MovieLens dataset we used was created using the real ratings of users in MovieLens. Pandora also started the Music Genome Project in 2000 for learning the similarities between music genres, artists, and songs to tailor recommendations. It was also around this time that Netflix started offering movie rentals online, using its proprietary Cinematch recommendation system to recommend movie rentals to customers. In 2006, Netflix hosted a competition for the best collaborative filtering algorithm that could improve on their own rating prediction system. This competition brought recommender systems to the spotlight, providing an avenue for the improvement and creation of new approaches to recommendations systems that could be applied across all domains. The winner of the competition used a model-

based approach, a type of collaborative filtering where a model is learned through machine learning for making predictions and recommendations. Since then, many different techniques have come to the forefront, but the main ideas of traditional collaborative filtering are still very relevant [3].

## III. Dataset

For our project, we chose the MovieLens dataset made available by GroupLens. The dataset was collected from MovieLens website over different periods of times. There are different sizes of datasets available based on the number of ratings by anonymous users who joined MovieLens in 2000. We choose the MovieLens 1M dataset which is a stable benchmark. The dataset is divided into several .csv files - users, ratings, movies and tags. The user table contains the user id, gender, occupation, age and zip code. The movie set stores information such as movie id, title and genre. The genre attribute is heavily used in content-based recommendation. The ratings are made on a 5-star scale. Each user has atleast 20 ratings. This helps overcome the cold start problem for new user.

$$\mathbf{tfidf}_{i,j} = \mathbf{tf}_{i,j} \times \log\left(\frac{\mathbf{N}}{\mathbf{df}_i}\right)$$

$\mathbf{tf}_{ij}$ = total number of occurences of i in j
$\mathbf{df}_i$ = total number of documents (speeches) containing i
N = total number of documents (speeches)

Fig. 1. tf-idf expression

### A. Content Based Filtering

There are two types of collaborative filtering: memory-based and model-based. Memory-based is the traditional approach to collaborative filtering, where similarities between users through ratings are exploited for making predictions of ratings and recommendations. On the other hand, model-based collaborative filtering uses models learned by deep learning or matrix factorization to find latent factors that can be thought of as underlying descriptors of the domain such as the amount of violence in a movie. Regardless of which method is used, the main goal of collaborative filtering is to learn a user's taste and preference by matching a target user to a large database of other users for whom preferences have been obtained in the past. To quantify preferences, ratings can be used - for example, the 1-to-5 scale like on MovieLens or "like" and "dislike" as done on YouTube. The rating is assumed to be an accurate representation of a user's feelings about a particular entity and a pool of these ratings can be leveraged by collaborative filtering algorithms to recommend items or services to users. Collaborative filtering handles the problem with content-based filtering by broadening the scope of recommendations since dissimilar items may be recommended.

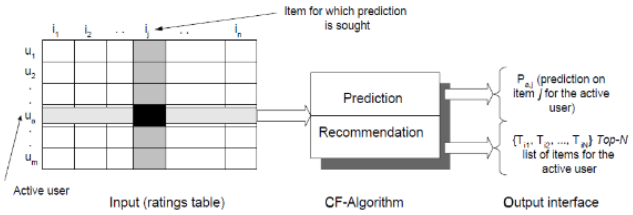


Fig. 2. Collaborative filtering
Source: http://www10.org/cdrom/papers/519/node6.html

*1) Memory-Based Collaborative Filtering:* There are two types of memory-based collaborative filtering: user-user collaborative filtering and item-item collaborative filtering. The main input is a user-item rating matrix. Our first implementation was based on user-user collaborative filtering. The premise behind user-user collaborative filtering is that there are clusters of users with similar preferences. Thus, if any group of users' rate items similarly, they will likely enjoy many other items mutually. For example, if two users both liked Star Wars and User 1 also liked The Lord of the Rings(LOTR), a user-user collaborative filtering approach is likely to recommend LOTR to User 2. Since the basis of any collaborative filtering algorithm is that similar users like similar items, there needs to be a metric to measure similarity between users for finding similar items to recommend. There are few similarity metrics that are used in collaborative filtering. One is the Jaccard measure that measures the number of common attributes two items have. For example, in the case of movies, the Jaccard measure of two movies could be defined as the number of common actors/actresses divided by the total number of unique actors/actresses in the two movies. Another measure, the cosine similarity measure, was already touched upon in the section on content-based filtering. For our purposes, we used the Pearson correlation. Using the Pearson correlation, the similarity between two users is measured from a scale of -1 to +1. A positive high

value indicates a high correlation, a negative high value indicates inversely high correlation, and a zero correlation indicates no correlation.

$$s(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (r_{ui} - \bar{r}_i)(r_{vi} - \bar{r}_i)}{\sqrt{\sum_{i \in \mathcal{I}_u} (r_{ui} - \bar{r}_i)^2 \sum_{i \in \mathcal{I}_v} (r_{vi} - \bar{r}_i)^2}}$$

Fig. 3. Pearson correlation

Here, $s(u,v)$ is the similarity between two users. $I_{uv}$ represents the entire set of items rated by both user $u$ and $v$. $r_{ui}$ represents the rating of item $i$ by user $u$. $r_{vi}$ represents the rating of item $i$ by user $v$ represents the average rating of user i across all items. Using scikit-learn's pairwise distances method, we created a user-user similarity matrix of size #users by #users where each entry in the matrix is the Pearson correlation between two pairs of users. Once the similarity matrix is computed, we can start making the recommendations. For a target user, we find the top 100 correlated users. Using all correlated users is impractical for large user databases and including users with low correlation will damage the recommendation quality. With the most similar users calculated, the last step is to predict the rating for items that the similar users have rated but the target user has not. The equation shown next uses the weighted average of all available ratings where the weights are the correlation values calculated previously between users. It is assumed all users rate using the same statistical distribution.

$$\hat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|}$$

Fig. 4. Statistical distribution

Here, $N_i(u)$ is the set of users who have rated item $i$. $r_{vi}$ is the rating given to item $i$ by user $v$ and $w_{uv}$ is the Pearson correlation between user $u$ and user $v$. Once the predicted rating user u would give items with no ratings, the top-N highest ratings can be returned for recommendations with N dependent on the context and use case of the application. If there is any drawback to user-user collaborative filtering, it is that many databases have more users than items, so it does not scale well. For example, Netflix has millions of users with many joining daily, but the number of movies and shows grows at a much smaller rate than the number of users. Thus, the dimensionality of the user space can become enormous, leading to difficulty in search for similarities due to the curse of dimensionality. Item-item collaborative filtering can mitigate this issue by working in the space of items. The idea is very similar to user-user collaborative filtering except item-item collaborative filtering searches for items rated similarly by various users. This is based on the premise that if many users rated a few items similarly, those items are likely similar and recommendation worthy. The equations used to calculate correlations between items and predict the ratings are nearly identical with just a few changes.

$$s(i,j) = \frac{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)(r_{ui} - \bar{r}_j)}{\sqrt{\sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_i)^2 \sum_{u \in \mathcal{U}_{ij}} (r_{ui} - \bar{r}_j)^2}}$$

Fig. 5. Equation for calculating the item-item similarity matrix which would be a #items by #items dimension matrix and each entry would be the Pearson correlation between two items.

In the equation above, $s(i,j)$ represents the similarity between two items, $i$ and $j$. $U_{ij}$ represents the set of users who rated both items $i$ and $j$. Like in user-based collaborative filtering, $r_{ui}$ is the rating user $u$ gave item $i$. Lastly, and are the average rating of items $i$ and $j$ across all users who rated those items. Thus, our approach to item-item collaborative filtering was very similar and we followed the same steps to create an item-item similarity matrix and predict the ratings using a weighted average as before.

In the equation above, $w_{ij}$ represents the correlation between item $i$ and item $j$. $N_u(i)$ is the set of items that users who rated item i have also rated. $r_{ju}$ is the rating given to item $j$ by user $u$. Once again, once

$$\hat{r}_{ui} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij} r_{ju}}{\sum_{v \in \mathcal{N}_u(i)} |w_{ij}|}$$

Fig. 6. User rating

the unknown ratings are predicted, the top-N items can be recommended. Both user-user collaborative filtering and item-item collaborative filtering follow the same basic framework, summarized below.

**Input**: User-Item Rating matrix $R$
**Output**: Recommendation lists of size $l$
**Const** $l$: Number of items to recommended to user $u$
**Const** $v$: Maximum number of users in $\mathcal{N}(u)$, the neighbours of user $u$
**For** each user $u$ **Do**
  **Set** $\mathcal{N}(u)$ to the $v$ users most similar to user $u$
  **For** each item $i$ that user $u$ has not rated **Do**
    Combine ratings given to item $i$ by neighbours $\mathcal{N}_i(u)$
  **End**
  Recommend to user $u$ the top-$l$ items with the highest predicted rating $\hat{r}_{ui}$
**End**

Fig. 7. Collaborative Filtering Pseudocode.
Source: "An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals"

The main criticisms to memory-based collaborative filtering are that it does not address the cold-start problem and struggles with sparsity. The sparsity arises from the fact that there may be many users in a database, orders of magnitudes more than the number of items, leading to a challenge of finding users who have rated items similarly. This is where model-based collaborative filtering can lend a helping hand.

*2) Model-Based Collaborative Filtering:* The reason memory-based collaborative is called memory-based is because the entire database of user-item ratings is located in memory in order to make recommendations. On the other hand, a model-based algorithm uses a model derived or learned to make recommendations. The model is learned off-line with the recommendation being generated on-the-fly whenever it is needed. As mentioned before, the Netflix competition of 2006 brought model-based approaches into the fray as the winner used an ensemble of over 100 models. One of the most common model-based approaches is to use matrix factorization where the user-item matrix is factored into two matrices which may reveal latent factors of the rating. These latent factors will be discussed shortly. Although matrix factorization is the most popular model-based approach, we decided to try a deep learning model for making recommendations. Thus, our goal was to learn the latent factors instead of deriving them. These latent factors can be described as embeddings which are low-dimensional hidden factors for both movies and users. For illustration purposes, lets assume 3-dimensional embeddings for both movies and users. The 3 numbers in the embedding matrix for a specific movie could be how recent the movie is, how many special effects are in the movie, and how CGI-driven a movie is. The embeddings for a user could be how much a user likes comedies, how likely is a user to give high ratings, and how often does a user watch movies. The image below shows how we create a user embedding matrix and a movie embedding matrix for learning the hidden factors. In a deep-learning approach, the embedding matrices would be learned rather than being derived as in matrix factorization.

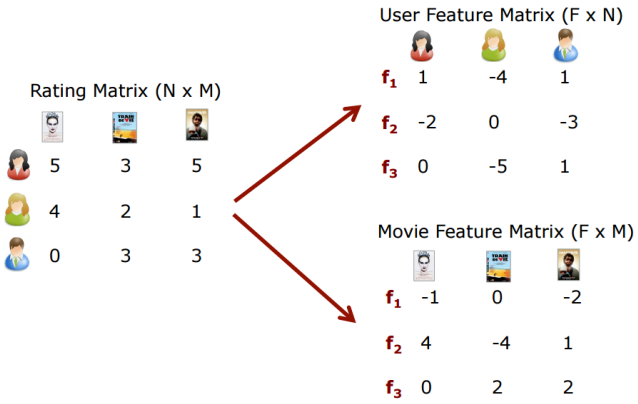


Fig. 8. Embedded Matrix Derived From User-Item Rating Matrix.
Source: towardsdatascience.com

In a deep-learning approach, the embedding matrices would be learned rather than derived as in matrix factorization. To learn these embeddings, we used Keras to create a deep learning model found to work well in literature.

The model consists of two embedding layers: a left embedding layer that will create the users embedding matrix and a right embedding layer that will create the movies embedding matrix. The input to the left embedding layer will be a user id and the input to the right embedding layer will be a movie id, and the latent factors for the user and movie will be returned. The final merge layer takes the dot product of the two latent

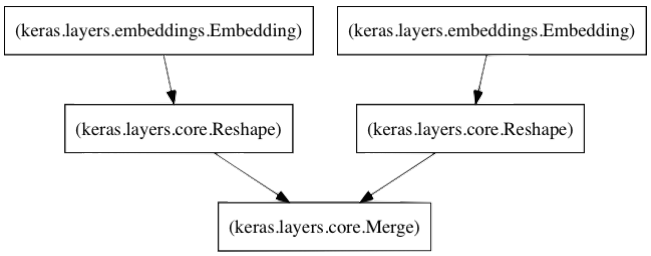


Fig. 9. Deep learning model used

factor feature vectors to produce a rating. If there are any drawbacks to model-based collaborative filtering, it is that the model needs to be continuously recomputed whenever data changes such as a user entering a rating.

## IV. Results

To evaluate our approaches for memory-based collaborative filtering, we divided the 1 million available ratings into two matrices: a training matrix that contained 80% of the ratings and a testing matrix that contained 20% of the ratings. We created the similarity matrix by using the training matrix and estimated the quality of the recommender system approaches on the test matrix where no ratings were used to create the similarity matrix. We used the root mean squared error to assess the quality.

$$RMSE(r, \hat{r}) = \sqrt{\frac{\sum_{a \in \mathcal{N}_{ui}} (r_a - \hat{r}_a)^2}{\|r\|}}$$

Fig. 10. Root-mean-square error

| Method | Training RMSE | Testing RMSE |
|---|---|---|
| Item-Item CF | 2.53 | 3.53 |
| User-User CF | 2.05 | 3.09 |

TABLE I

THE RESULTS ON BOTH THE TRAINING DATA AND TESTING DATA

As you can see, user-user collaborative filtering has the lower RMSE on both the training data and test data. To train the model in Keras, we trained for 30 epochs on the training matrix from before that contained 80% of the ratings, using Adam gradient descent with Early Stopping which stops training whenever the validation error does not decrease for 4 epochs in a row. The validation set was the same testing matrix used in the memory-based collaborative filtering approaches.

The best validation RMSE we had was 0.7494 and we used the weights that generated that error for the final model. The RMSE for deep learning is lower than the RMSE for both user-user collaborative filtering and item-item collaborative filtering. This is in accordance with contemporary results where model-based approaches tend to perform better than memory-based approaches.

```
                                    Tigger Movie, The (2000)
                 We're Back! A Dinosaur's Story (1993)
                              Road to El Dorado, The (2000)
                                          Dinosaur (2000)
                           Pokémon the Movie 2000 (2000)
                                        Toy Story (1995)
                      Aladdin and the King of Thieves (1996)
                                   American Tail, An (1986)
               American Tail: Fievel Goes West, An (1991)
                                Rugrats Movie, The (1998)
                                     Bug's Life, A (1998)
                                       Toy Story 2 (1999)
                                    Saludos Amigos (1943)
                                       Chicken Run (2000)
Adventures of Rocky and Bullwinkle, The (2000)
                                    Goofy Movie, A (1995)
                                  Gulliver's Travels (1939)
                                  Digimon: The Movie (2000)
                                      Lion King, The (1994)
                 Snow White and the Seven Dwarfs (1937)
```

Fig. 11. Content-based recommendation for the movie "The Sword in the Stone (1960)"

| title | prediction | genres |
|---|---|---|
| Usual Suspects, The (1995) | 4.849084 | Crime\|Thriller |
| Silence of the Lambs, The (1991) | 4.689820 | Drama\|Thriller |
| Pulp Fiction (1994) | 4.577641 | Crime\|Drama |
| One Flew Over the Cuckoo's Nest (1975) | 4.479474 | Drama |
| Seven (Se7en) (1995) | 4.426721 | Crime\|Thriller |
| Great Escape, The (1963) | 4.346505 | Adventure\|War |
| GoodFellas (1990) | 4.332287 | Crime\|Drama |
| Saving Private Ryan (1998) | 4.327280 | Action\|Drama\|War |
| Pather Panchali (1955) | 4.309986 | Drama |
| Sanjuro (1962) | 4.288775 | Action\|Adventure |
| Animal House (1978) | 4.246664 | Comedy |
| Game, The (1997) | 4.224116 | Mystery\|Thriller |
| Sling Blade (1996) | 4.223121 | Drama\|Thriller |
| American History X (1998) | 4.208044 | Drama |

Fig. 12. Item-item collaborative filtering recommendations for a random user

| title | prediction | genres |
|---|---|---|
| Patriot, The (2000) | 4.648333 | Action\|Drama\|War |
| Pather Panchali (1955) | 4.636128 | Drama |
| Remember the Titans (2000) | 4.611228 | Drama |
| Bone Collector, The (1999) | 4.487613 | Thriller |
| Chushingura (1962) | 4.483106 | Drama |
| King of Masks, The (Bian Lian) (1996) | 4.453830 | Drama |
| Green Mile, The (1999) | 4.421598 | Drama\|Thriller |
| Aparajito (1956) | 4.396999 | Drama |
| Time to Kill, A (1996) | 4.359523 | Drama |
| Armageddon (1998) | 4.358553 | Action\|Adventure\|Sci-Fi\|Thriller |
| Children of Heaven, The (Bacheha-Ye Aseman) (1... | 4.352197 | Drama |
| Microcosmos (Microcosmos: Le peuple de l'herbe... | 4.336449 | Documentary |
| Perfect Blue (1997) | 4.335655 | Animation\|Mystery |
| Sanjuro (1962) | 4.329998 | Action\|Adventure |

Fig. 13. User-user collaborative filtering recommendations for a random user

| title | prediction | genres |
|---|---|---|
| Saving Private Ryan (1998) | 4.912568 | Action\|Drama\|War |
| Silence of the Lambs, The (1991) | 4.859630 | Drama\|Thriller |
| GoodFellas (1990) | 4.846872 | Crime\|Drama |
| Gladiator (2000) | 4.499942 | Action\|Drama |
| Usual Suspects, The (1995) | 4.363734 | Crime\|Thriller |
| Platoon (1986) | 4.342752 | Drama\|War |
| Seven (Se7en) (1995) | 4.334940 | Crime\|Thriller |
| American History X (1998) | 4.290866 | Drama |
| Pulp Fiction (1994) | 4.259939 | Crime\|Drama |
| Great Escape, The (1963) | 4.243103 | Adventure\|War |
| Predator (1987) | 4.226206 | Action\|Sci-Fi\|Thriller |
| Green Mile, The (1999) | 4.171514 | Drama\|Thriller |
| Caddyshack (1980) | 4.171106 | Comedy |
| Glory (1989) | 4.170219 | Action\|Drama\|War |

Fig. 14. Deep learning embeddings on the user-movie matrix

## V. Future Work

We have worked on providing recommendations using content-based filtering, collaborative filtering and deep learning. The embeddings from deep learning provide promising results with high accuracy. In the future, we want to try hybrid models mainly using a layer of LSTM to capture the relationship of the ratings done by the user recently or to use a mixture of content-based and collaborative filtering for solving the cold start problem. We also intend to work on solving the sparsity problem of collaborative filtering technique, mainly allow multiple matrices for different criteria such as native location, age and occupation.

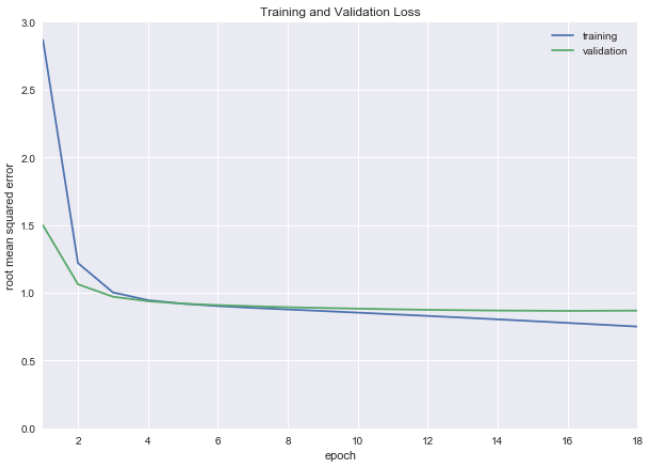


Fig. 15. Training and validation loss

## VI. Conclusion

Movie recommendations is an emerging field of interest among the media industries such as Netflix, Disney, Amazon and Hulu. There is a constant need to improve the recommendations provided to the users. Due to business requirements, companies do not open-source the dataset. In this scenario, we have used the MovieLens to understand and provide recommendations using collaborative filtering, content-based filtering and deep learning. We tested our model for random users and measured the predication accuracies among our recommendations. The high accuracy of our results has reaffirmed confidence in the above methods.

## References


[1] Desrosiers, Christian, Karypis, George. 2011. "A comprehensive survey of neighborhood-based recommendation methods". University of Minnesota

[2] Harper, Maxwell F. and Konstan, Joseph. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

[3] Herlocker, Jonathan L. et al. "Evaluating collaborative filtering recommender systems." ACM Trans. Inf. Syst. 22 (2004): 5-53.

[4] Levinas, Adrian Claudio. 2014. "An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals"

[5] Linden, Greg et al. "Industry Report: Amazon.com Recommendations: Item-to-Item Collaborative Filtering." IEEE Distributed Systems Online 4 (2003): n. pag.

[6] Resnick, Paul et al. "GroupLens: An Open Architecture for Collaborative Filtering of Netnews." CSCW (1994).

[7] Schafer, J. Ben et al. "Collaborative Filtering Recommender Systems." The Adaptive Web (2007).

[8] Wei, Jian et al. ”Collaborative filtering and deep learning based recommendation system for cold start items.” Expert Syst. Appl. 69 (2017): 29-39.
[9] http://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0
[10] http://link.springer.com/chapter/10.1007/978-3-642-28466-3_58
[11] http://www.gregreda.com/2013/10/26/using-pandas-on-the-movielens-dataset/
[12] http://blog.statsbot.co/recommendation-system-algorithms-ba67f39ac9a3
[13] http://www10.org/cdrom/papers/519/