Movie Recommender System

	NetID	Question asked to (Group 6)	Question answered from (name)
Liyong Cui	lcui9	Group 17	Armaan Mohajir
SUBHRA KANUNGO	skanun2	1.Group 8 2. Group 3	Armaan Mohajir
Huihan Hu	hhu25	Group 18	Srishti Pyasi
Filmon Ghebremeskel	fghebr2	Group 7	

1 Introduction & Related Works

The rapid growth of active Internet population in 21st century has a great impact on people's daily life.[1] It has reshaped the modern living in all aspects. The Internet offers to everyone the opportunity of sharing and collecting information about every possible topic. In the meanwhile, people are facing the difficulty on how to take advantage of the enormous volume of information, and try to identify the data needed in the reasonable amount of time.[2] The rising challenge drives the need for assistance to distinguish as fast as possible the most suitable choices based on their own preferences. Surprisingly, such needs are not only arise in research or other pro- fessional activities, like the search of scientific literature, but also in the field across industries including healthcare, entertainment, and retail. In the health industry, bio-informatics compa- nies start using patient health indices to monitor, predict, and provide early-stage intervention to prevent potential diseases and health decline.[3] In the entertainment industry, companies such as YouTube and Netflix are using user tendencies and interests in watched entertainment to predict potential movies/TV shows of interest.[4] In e-commerce, companies like Amazon are using predictive analytic to promote and recommend products based off of customer interests.[5] How can users find new compelling new product? Yes, one can use search to access content. However, a recommendation engine can display items that users might not have thought to search for on their own. Clearly, motivation for using predictive analysis at companies like Netflix, Hulu, Facebook, twitter, shopify, and Amazon are mainly for maximizing business ef- ficiency. The predicted information by analyzing the retrieved user inputs can forecast sales, create long term customer/consumer relationships, improve user experience, and ultimately save consumers and customers time and money but increase cooperation revenue as well. This is why recommending systems have been developed to provide customized information, such as web pages, restaurants, TV programs, movies and so on. These systems are based on modeling content, as well as user preferences and social groups, to provide good recommendations.

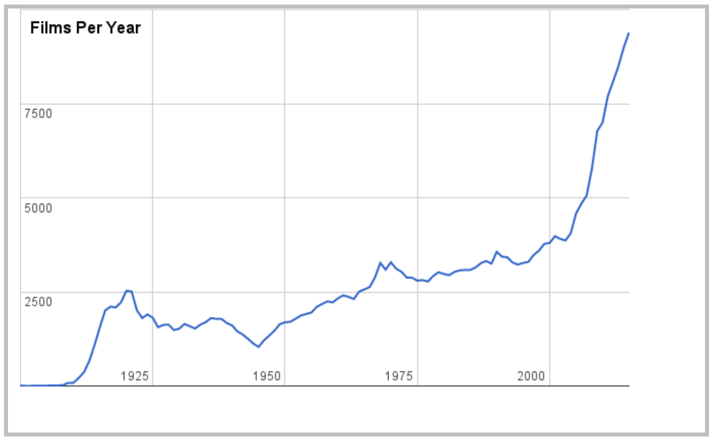

Figure 1: Films per year from IMDB database

As far as movie recommendation are concerned, the problem of selecting a nice movie is getting more and more difficult as time passes. According to IMDB[6], since 1900 there's been an average of 2,577 films produced each year around the world. But that doesn't really capture the amazing growth taking place in the industry right now. As one can easily appreciate in Figure 1, starting form the new millennium, we saw a sonic boom in film production, starting a trend of growth that has never stopped or even slowed. As a result, to help users find shows and movies to enjoy with minimum effort is becoming more centralized service in media streaming industry. According to Netflix, the platform will estimate the likelihood that one will watch a particular title in their catalog based on a number of factors: 1) users' viewing history and how they rate other titles; 2) other members with similar tastes and preferences on the streaming platform; 3) information about the titles, such as their genre, categories, actors, release year etc. In addition, Netflix will also track things like: the time of day an individual user watch, the devices you are watch Netflix on, and how long one watch in order to give the best personalized recommendations.

Recommender systems are based on a variety of approaches such as content-based filtering[7, 8], collaborative filtering[9, 10, 11], and hybrid method[12, 13, 14]. The idea behind Content- based recommendation system is to recommend items similar to what the user likes, based on their previous actions or explicit feedback. In simple words, one may get recommendation for a movie based on the description of other movies, which have been watched in the history. On the other hand, the theory behind collaborative filtering is to work with collaboration with user or movie id. For example, there are two user A and B, user A likes movie 1, 2, 3, and 4 and user B like movies 2, 3, 4, and 5. Since movies 2, 3 and 4 are similar to both user, therefore, movie 1 will be recommended to user B and movie 5 will be recommended to used A. Each of these techniques has its own strengths and weaknesses. In search of better performance, researchers have also combined recommendation techniques to build hybrid recommender systems. These methods can also be used to overcome some of the common problems in recommender systems such as cold start and the sparsity problem, as well as the knowledge engineering bottleneck in knowledge-based approaches. Netflix is a good example of the use of hybrid recommender systems.[14] The website makes recommendations by comparing the watching and searching habits of similar users (i.e. collaborative filtering) as well as by offering movies that share characteristics with films that a user has rated highly (i.e. content-based filtering).

2 Material and Methods

2.1 Description of data sets

The data set contains movies graded by 12 users. The rating consists of a decimal number on a 1-5 scale assigned to each movie, where 1 is the lowest grade (i.e. ”Do not like at all”) and 5 is the highest grade (i.e. ”Really like it”). Missing values are present when a user has not watched the corresponding movie yet.

Based on this data set, in the next section, two methods have been implemented and discussed. 1) Item-based (or memory-based) collaborative filtering, and 2) model-based collaborative filtering using matrix factorization have been implemented on group movie rating dataset.

2.2 Item-based Collaborative Filtering

2.2.1 Method

The first method that was implemented to recommend the user new movies is based on ”Item- based collaborative filtering”. This systems recognizes similar movies based on the previous users’ ratings. For example, if users A, B, and C watch movies X and Y and rate them 5 stars, when user D watches movie X, he will get a recommendation to watch movie Y as well, being rated 5 stars as well by user who watched both X and Y. The idea behind collaborative filtering is that similar users share the same interest and that similar items are liked by similar users.

In collaborative filtering (CF), the matrix that is being used is called utility matrix. Collaborative filtering is trying to fill out the missing values in the matrix based on what the non missing values tell about the user or the movie considered. The non-missing values, in this particular project’s case, are called ”explicit opinions”, because they explicitly point out how much a user liked a movie.

In CF, a measure of similarity between either users or movies is used. The similarity measure is defined by Pearson correlation or cosine similarity, defined as follows:

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \qquad \cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

where $u_{i,k}$ could either be the similarity between user i and user k or between movie i and movie k. Specifically, in item based CF, $u_{i,k}$ is the similarity between movie i and movie k, $v_{i,j}$ is the rating given by user j to movie i (which is the unknown variable for those movies that were not rated), v_i is the mean rating given to movie i, and the other variables can similarly be interpreted. Both formulas can be used, but in this project Pearson similarity was used being invariant to adding a constant to all elements. The user opinion can be extracted by the following formula:

$$v_{ij}^* = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj}$$

where K is a multiplier constant, v_{ij} is the missing rating.

Item based CF avoids the problem of not being able to get good results if a user has a changing preference of a certain movie over time (dynamic user preference). The main issue that remains, though, is scalability, meaning that the computation grows along with the users and the movies. The complexity can get to $O(mn)$ with m users and n movies. Sparsity can be another problem because movies could be rated very high just because they have a similar rank if there is just a singular user who ranked them both.

2.2.1 Results and Discussion

It was possible to calculate the similarity of the other movies to Forrest Gump based on the users who rated both Forrest Gump and other movies. The movies that received a high rating by a user who rated Forrest Gump high as well, are more correlated to Forrest Gump. The results below where obtained:

```
Frozen I                          -0.172579
Pirates of the Carribean I        -0.491952
Harry Potter series                0.600024
Titanic                            0.732023
The GodFather series                    NaN
dtype: float64
```

Reasonably, Forrest Gump is the most correlated movie to itself. Black Panther appears to be the most recommended movie if the user has watched Forrest Gump, and following that The Notebook, Pulp Fiction and so on.

In order to avoid non reliable recommendations, and due to the limits mentioned above related to Item Based CF, a threshold was set on the first step to consider only movies rated by 3 or more users. This avoids the recommendation of movies with a high rating similar to Forrest Gump but only rated by a few users, which is a common error when using Item based collaborative filtering. Additionally, the neighborhood method was used to calculate the user-to-user similarity in taste. Final predictions of the missing of the missing values were also performed.

User-to-user similarity matrix was obtained:

```
          Rogerio Lauren Shreya Daiqing   Shin Songyun    Mo Dr. Leow Akanksha
Rogerio        NA  0.568  0.410   0.538  0.555   0.788 0.737    0.337    0.422
Lauren      0.568     NA  0.824   0.524  0.533   0.243 0.319    0.020    0.260
Shreya      0.410  0.824     NA   0.591 -0.046   0.228 0.616   -0.113    0.411
Daiqing     0.538  0.524  0.591      NA  0.436   0.436 0.648    0.562    0.543
Shin        0.555  0.533 -0.046   0.436     NA   0.312 0.618    0.227    0.412
Songyun     0.788  0.243  0.228   0.436  0.312      NA 0.735    0.325    0.595
Mo          0.737  0.319  0.616   0.648  0.618   0.735    NA    0.345    0.731
Dr. Leow    0.337  0.020 -0.113   0.562  0.227   0.325 0.345       NA    0.135
Akanksha    0.422  0.260  0.411   0.543  0.412   0.595 0.731    0.135       NA
Marco       0.357  0.316  0.083  -0.286  0.454   0.411 0.264   -0.486    0.230
Liyong      0.244  0.458  0.754   0.385  0.464  -0.654 0.640    0.403    0.837
Federica    0.653  0.747  0.500   0.327  0.427   0.504 0.714    0.237    0.481
```

According to the above table, user Akanksha and Liyong seem to have the most similar tastes, being their similarity the maximum value in the table, equal to 0.8367420. Reversely, Liyong and Songyun have the least similar taste in movies, with a similarity value equal to -0.65370792. Also, it can be noticed that roughly most of the users have a different taste to that of Dr. Leow, being the similarity values often small or negative. The rating predictions were then made, based on the weighted neighborhood avaraged with the similarity weights. New matrixes were obtained with no missing values and the missing values were replaced by predictions. Partial dataset with predicted ratings is shown as below:

```
         The.Martian  Arrival The.Lego.Movie Moonlight  Get.out La.La.Land
Rogerio     2.000000 5.000000       3.000000  1.000000 4.500000   4.000000
Lauren      4.000000 4.000000       3.930713  3.754497 4.085063   4.000000
Shreya      4.000000 4.257182       4.000000  3.751504 4.245808   5.000000
Daiqing     3.916284 4.535539       3.950596  3.491787 4.145645   5.000000
Shin        4.500000 5.000000       4.000000  4.500000 4.000000   5.000000
Songyun     3.796591 4.000000       3.882323  3.397272 4.000000   4.303474
Mo          3.683669 4.000000       3.743884  2.000000 3.974581   5.000000
Dr. Leow    2.866342 5.000000       2.992781  2.198908 3.302946   3.000000
Akanksha    3.812798 4.116585       3.842009  3.526506 3.957798   3.500000
Marco       3.675583 3.977730       3.683830  3.160952 4.072868   4.584397
Liyong      4.036180 4.438374       4.025566  3.743159 4.000000   4.000000
Federica    3.876933 4.426353       3.932477  3.415765 4.168633   4.500000
```

2.3 Matrix Factorization

2.3.1 Method

We approached matrix factorization as another recommender algorithm. Matrix factorization is a collaborative filtering model which has the advantage to allow to discover the latent features underlying the interactions between users and items.
The data set in the problem is in the form of a sparsely filled matrix, with each user representing a row and each movie a column, so that each cell in the matrix either contains

an observed rating (1-5) for a single movie by a single user, or is blank, meaning the user did not see that film. The goal is to predict how the users would rate the blank items: on the base of these predictions, then, recommendation to the users are made.
For doing this, the user-film rating matrix is approximated by compressing its information into two lower dimensional matrices: the intuition behind this is that, if there are some latent features that determine how a user rates an item, then there must exist a more concise and descriptive way to represent the information than a matrix containing completely independent and unrelated ratings [15]. For instance, any given movie can be described in terms of attributes such as whether it's an action movie, a comedy or a drama, what actors are in it, and so on. In the same way, every user ratings can be described in terms of whether they prefer action movies, comedies or dramas, what stars they like, and so on. Hence, if we can discover these latent features, we are able to predict a rating with respect to a certain user and a certain item: if a user likes a given actor, and a film has that actor in it, this will contribute positively on his rating to that film while if the film is a drama but the subject tends to give lower score to dramas, this will contribute negatively to his rating. This means that, to obtain a single rating starting from the latent features, it is only needed to multiply each user preference by the corresponding movie aspect, and then sum all the products up into a final opinion.
In matrix terms, the user-film rating matrix R can be approximated by the decomposition into two oblong matrices P and Q such that P has a row for each user and a column for each latent feature and Q has a row for each film and, again, a column for each latent feature. In this way, each row of P represents the strength of the associations between a user and the features. Similarly, each row of Q represents the strength of the associations between a film and the features. This concept is graphically shown in Fig.2.

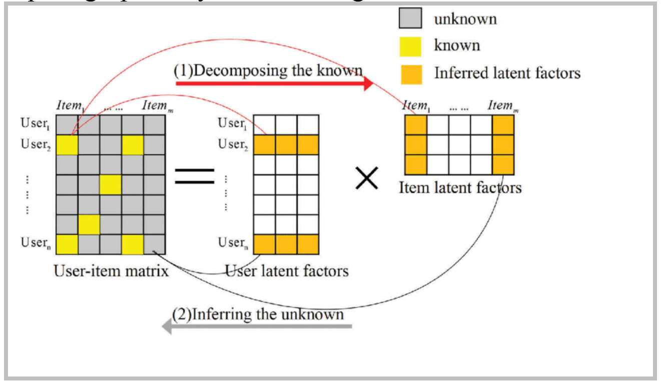

Figure 2: Diagram of matrix factorization [16]. The user latent factor matrix is what we called P. The Item latent factors matrix is what we called Q^T. User-item matrix is R.

Rating predictions for each item can be found through $\hat{R} = P \times Q^T$ $\left(\hat{R}(i,j) = p_i q_j^T = \sum_{k=1}^{k} p_{ik} q_{ik}\right)$.This means that, given P and Q, the prediction of the i-th rater relative to the j-th film is equivalent to the dot product between the i-th row of P and the j-th row of Q. The parameter k (i.e. the number of latent feature) is used to tune the expressive power of the model. k was chosen based off of having the number of features smaller than the number of users as well as a low enough loss function. Moreover, the matrix R will be of rank k and, since k should be smaller than the number of users and the number of films, than R can be considered a lower rank approximation of R. It is not reasonable, in fact, to think that to each user can be associated a unique feature: in this situation there is no way to do recommendations [18].

2.3.2 Find P and Q using Gradient Descent

To find P and Q we used Gradient Descent (GD) joined with an implementation of alternating minimization of the error metric. This means that the optimization problem is solved only in the non-missing set: defining Ω as the set of non-missing observations in R, the following problem is solved

$$\min_{\forall(i,j)\subset\Omega}\sum_{i,j}(e_{ij})^2 = \min_{\forall(i,j)\subset\Omega}\sum_{i,j}(R(i,j)-\widehat{R}(i,j))^2$$

$$\widehat{R} = P\times Q^T \implies \widehat{R}(i,j) = p_i q_j^T$$

A regularization term is also added to the loss function to control the magnitudes of the user-feature and film-feature vectors such that large numbers in P and Q are penalized. We used the sum of the frobenius norms of P and Q weighted by the parameter β. This parameter is used to weight the amount of penalization desired. The quantity to be minimized becomes:

$$e_{ij}^2 + Reg_{ij} = (R(i,j)-\widehat{R}(i,j))^2 + \frac{\beta}{2}\sum_{k=1}^{k}(p_{ik}^2+q_{jk}^2)$$

$$\min_{\forall(i,j)\subset\Omega}\sum_{i,j}((e_{ij})^2+Reg_{ij}) = \min_{\forall(i,j)\subset\Omega}\sum_{i,j}(R(i,j)-\widehat{R}(i,j))^2 + \frac{\beta}{2}(||P||_F^2+||Q||_F^2)$$

$$= \min_{\forall(i,j)\subset\Omega}\sum_{i,j}(R(i,j)-\sum_{k=1}^{k}p_{ik}q_{jk})^2 + \frac{\beta}{2}(||P||_F^2+||Q||_F^2)$$

In accordance with GD algorithm, the update rule for finding P and Q will be

$$p_{ik}^{t+1} \leftarrow p_{ik}^t + \alpha\frac{\partial}{\partial p_{ik}}({e_{ij}^t}^2 + Reg_{ij}) = p_{ik}^t + \alpha(2e_{ij}^t q_{jk}^t - \beta p_{ik}^t)$$

$$q_{jk}^{t+1} \leftarrow q_{jk}^t + \alpha\frac{\partial}{\partial q_{jk}}({e_{ij}^t}^2 + Reg_{ij}) = q_{jk}^t + \alpha(2e_{ij}^t p_{ik}^t - \beta q_{jk}^t)$$

Where α is the learning rate, and t refers to the epoch of the algorithm. Two different versions of the algorithm with two different stop criterion were implemented. The first "unrefined" stop criterion tried, was based both on the value of the loss function and on the number of iterations: the algorithm stops iterating if the number of iterations reach 3000 or if our loss function is smaller than 0.001. The second "more refined" stop criterion that was implemented adds to the first one the possibility to stop iterating if the loss function does not have a percent decrease of at least 0.1%: in this case convergence was considered reached. After some trials, furthermore, we set the learning rate to α = 0.02 and the amount of regularization β = 0.00005.

$$LOSS = \min_{\forall(i,j)\subset\Omega}\sum_{i,j}((e_{ij})^2+Reg_{ij}) = \min_{\forall(i,j)\subset\Omega}\sum_{i,j}(R(i,j)-\widehat{R}(i,j))^2 + \frac{\beta}{2}(||P||_F^2+||Q||_F^2)$$

Again, movies with few ratings, less than 4, were not included in the prediction model because the latent features relative to them would not have been a reliable representation of the film.

2.3.3 Results

After eliminating movies with fewer than 4 ratings, there were 42 movies and 12 raters, with a total of 298 ratings (206 un-rated rater-movie pairs). A histogram of the number of ratings per movie is shown in Fig. 3. The ratings were left-skewed with a mean rating of 3.9. The distribution of movie ratings is shown in Fig. 4.
Loss function history during Gradient Descent for numbers of latent features (k) between 2 and 10 is shown in Fig.5. For k ≤ 5 the algorithm was not able to converge and for k ≥ 6 the algorithm was always able to converge. Fig.6 shows the final value of the loss function (relative to the final matrix factorization found by GD) for numbers of latent features between 6 and 10. Using the elbow method in the loss function vs. k results, 8 latent features was determined to be optimal. Both Fig.5 and Fig.6 were obtained with the "STOP CRITERION": Iteration Number > 3000 or LOSS < 0.001. However, since the dataset used is quite small, it is likely this is due to overfitting to the existing ratings. For this reason, cross validation and hyper parameter tuning is implemented.

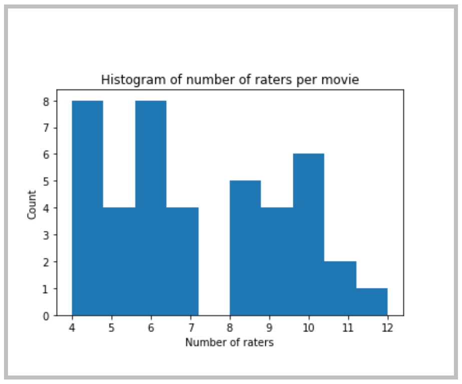

Figure 3: Histogram of number of raters per movie (42 movies total, and yet Hitchhikers Guide to the Galaxy is not one of them)

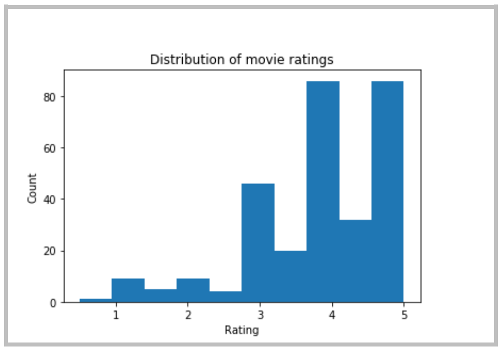

Figure 4: Histogram of ratings (1-5)

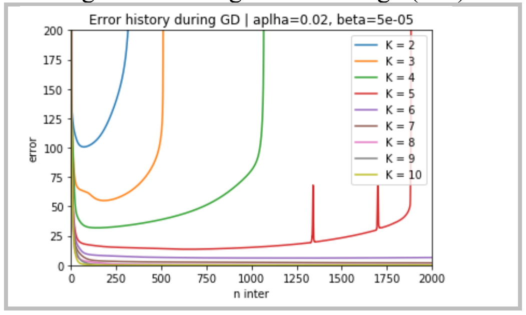

Figure 5: Loss function path with various k.

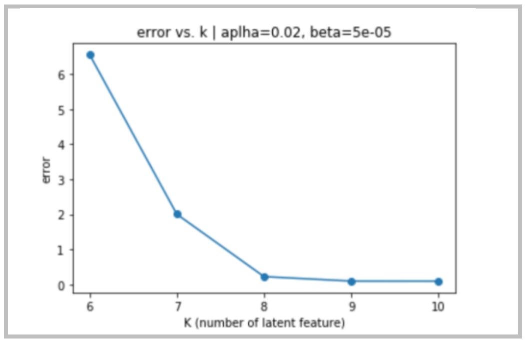

Figure 6: Loss function of the factorization obtained with different k: an “elbow” is individuated at k = 8.

In order to tune our model to avoid over-fitting, we implemented cross-validation with an exhaustive grid search of the hyper-parameters of K (number of latent features) and beta (regularization parameter). We thus utilize the matrix factorization with gradient descent algorithm with K's from 2:10 and beta from 0.01 to 0.1. With each pair of parameters, we perform 10-fold cross validation.
In order to perform cross-validation with matrix factorization, it is necessary to remove random rater-movie pairs (instead of removing one rater or one movie entirely), as matrix factorization requires the full set of raters and movies to achieve matrix completion. The accuracy of the model is then assessed on the values that were removed from input to matrix factorization. A heatmap of the MSE across K and b is shown in Figure 7. Finer tuning could be obtained by including extra repetitions of cross validation. However, from this tuning, it can be observed that over-fitting is appropriately controlled at beta=0.04, and achieves a minimum MSE of 0.8. A plot of the first two latent rater features is shown in Figure 8.

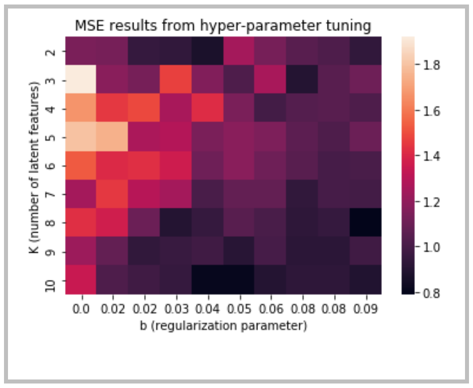

Figure 7: Loss Hyper-parameter tuning results

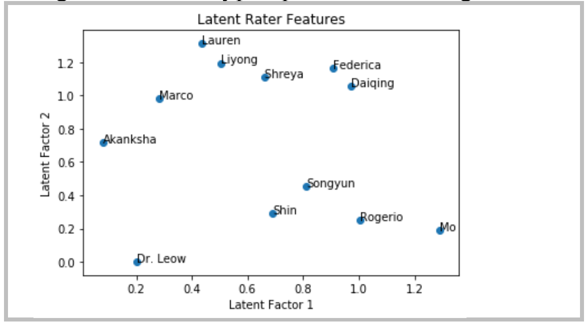

Figure 7: Principal latent rater factors

3 Summary

For the item-based collaborative filtering section, we are trying to predict the rating of a user for an film based on the ratings of the user for some similar films. In this respect, two films, which we call neighbors, are similar if several users of the system have rated these films in a similar fashion. One of the main advantages of neighborhood approaches is that it can capture local associations in the data. To provide a example, it is possible for a movie recommender system based on this type of approach to recommend the user a movie very different from his usual taste or a movie that is not well known, when one of his closest neighbors has given it a strong rating. This recommendation may not be a guaranteed success, as would be a romantic comedy, but it may help the user discover a whole new genre or a new favorite actor/director. This unexpected, pleasant discovery is called "serendipity", which is playing an important role in the appreciation of users for the recommender system. In addition, neighborhood approaches are intuitive and relatively simple to implement. They are also stable with constant addition of users, items and ratings but they are known to be sensitive to sparseness of ratings [17].

In section 2.3, we discussed one latent factor model for building collaborative filtering recommenders: matrix factorization (e.g., Singular Value Decomposition (SVD)). The method transforms both items and users to the same latent factor space. Then the latent space is used to explain ratings by characterizing both films and users in term of factors automatically inferred from user feedback. Nevertheless, conventional SVD method is undefined when knowl- edge about the matrix is incomplete. Moreover, carelessly addressing only the relatively few known ratings is highly subjective to overfitting. A more advanced ”SVD++ model” will be introduced in the next part to increase prediction accuracy by integrating other sources of user feedback[18].

References

[1] http://www.internetworldstats.com/stats.htm .

[2] Montaner, M., LÃ 3pez, B., DeLaRosa, J.L.(2003).A taxonomy of recommender agents on the internet. 330.

[3] Mehta, N., Pandit, A. (2018). Concurrence of big data analytics and healthcare: A systematic review. International journal of medical informatics, 114, 57-65.

[4] Daniel Faggella (2017). The ROI of recommendation engines for marketing. https://martechtoday.com/roi-recommendation-engines-marketing-205787

[5] Corinna Underwood (2020). Use Cases of Recommendation Systems in Business ˆa Current Applications and Methods. https://emerj.com/ai-sector-overviews/use-cases-recommendation-systems/

[6] IMDB.com. ”Press Room”. Retrieved October 5, 2018.

[7] Aggarwal, Charu C. (2016). Recommender Systems: The Textbook. Springer. ISBN 9783319296579

[8] D.H. Wang, Y.C. Liang, D.Xu, X.Y. Feng, R.C. Guan. (2018) A content-based recommender system for computer science publications. Knowledge-Based Systems, 157: 1-9

[9] Sarwar, Badrul, George Karypis, Joseph Konstan, and John Riedl. (2001) Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web, pp. 285-295.

[10] Linden, G., Smith, B., York, J. (2003) Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 7(1), 76-80.

[11] Breese, J. S., Heckerman, D., Kadie, C. (2013). Empirical analysis of predictive algorithms for collaborative filtering. arXiv preprint arXiv:1301.7363.

[12] Burke, R. (2007). Hybrid web recommender systems. In The adaptive web (pp. 377-408). Springer, Berlin, Heidelberg.

[13] Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. and Sampath, D. (2010). The YouTube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems (pp. 293-296).

[14] Gomez-Uribe, C. A., and Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), 1-19.

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

[16] Zhang, Shuai and Yao, Lina and Sun, Aixin and Tay, Yi Deep Learning Based Recommender System. Association for Computing Machinery (ACM), 2019.

[17] Cai L, Xu J., Liu J., Pei T. (2017). Integrating spatial and temporal contexts into a factor- ization model for POI recommendation. International Journal of Geographical

Information Science. 32. 1-23. 10.1080/13658816.2017.1400550.

[18] Ricci, Francesco and Rokach, Lior and Shapira, Bracha. (2015) Recommender systems: introduction and challenges. Recommender systems handbook. Springer, Boston, MA, 2015. 1-34.