CS550 - Movie Recommendation System

Jash Gaglani (jg1700)
Rutgers University
jash.gaglani@rutgers.edu

Janish Parikh (jrp328)
Rutgers University
janish.parikh@rutgers.edu

ABSTRACT

A movie recommendation is important in everyone's social life due to its prowess in providing enhanced entertainment.
Our goal is to develop a movie recommendation system based on the 'MovieLens' dataset. We strive to integrate the aspects of user's personal preference with the overall features of movie such as genre, popularity etc. We have implemented popularity model, collaborative filtering model and latent factor based model.The final recommendation list for each user is generated, by combine all the methods to form a hybrid model to overcome the limitations of individual models. The hybrid model performs better than the individual models in terms of quality and diversity of recommendations. Detailed analysis of each model is given in the report.

KEYWORDS

movie recommendation, content-base, collaborative filtering, latent factor, SVD, hybrid recommendation, Precision, RMSE, Recall, F-1 Score, Normalized Discounted Cumulative Gain or NDCG

1 INTRODUCTION

With the large amount of growth in data, and capturing the consumer behaviour, one of the most important aspects of today's technologies is providing personalized services. Offering personalized services is what sets off data-driven user service applications from the vanilla application.
We concentrate one such personalized service specifically, Recommendations system. They are used in variety of applications, and are the ones that are enabling the big techs like Amazon, Netflix, Disney, Hulu etc to aggrandize their profits. In the current era, where data generating and processing capacities is in the petabytes, we are entering into the next phase of recommendation systems. With such abundance of data about the user and the items, we can experiment with a lot of diverse and sophisticated techniques including deep learning methods to try and improve the user experience by providing better recommendations.
In this project we worked on the MovieLens-100k datasets to build our recommendation systems.

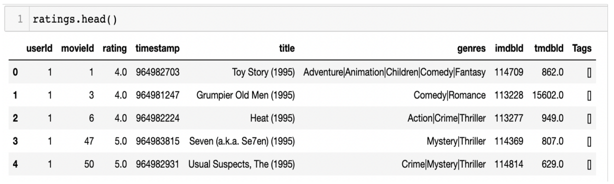
Conference'17, Washington, DC, USA
© 2016 ACM. 978-x-xxxx-xxxx-x/YY/MM...$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

Survey on recommendation system methods: This [1] paper covers different techniques which are used in recommendation system and also proposes a new system for efficient web page recommendation based on hybrid collaborative filtering i.e. using collaborative technique and CHARM algorithm which are coupled with the pattern discovery algorithms such as clustering and association rule mining.
Below are some of the related work done for recommendation systems.
Collaborative Filtering Recommender Systems: This paper [3] defines the primary uses for users of the adaptive web, the theory and practice of CF algorithms, and design decisions regarding rating systems and acquisition of ratings. It also discusses the ways on how to evaluate CF systems, and the evolution of rich interaction interfaces.
A personalised movie recommendation system based on collaborative filtering: This paper discusses the idea of how to make use of filtering and clustering techniques to suggest items of interest to users. The experiment results on the MovieLens dataset in this paper provide a reliable model which is precise and generates more personalised movie recommendations compared to other models.

3 DATA

The primary dataset used for this project is the grouplens movie review dataset. The smaller dataset (ml-latestsmall) describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. The dataset has a collection of 27,753,444 reviews over 58,098 different movies by 283,228 users. Each rating being a value between 0-5. Some features of the dataset include the timestamp of the review, genre of the movie, the keywords of comments by the users, and the IMDB and TMDB id for the corresponding movie.
We split the dataset into test and train on the basis of UserID,



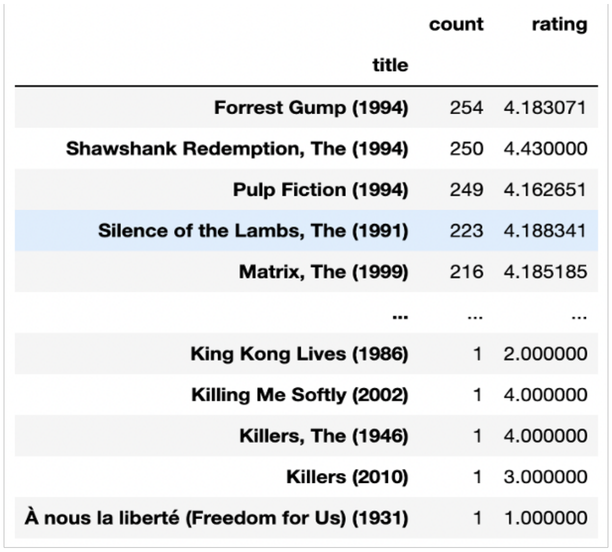
```
ratings.head()
```

```
   userId  movieId  rating  timestamp                         title                                       genres  imdbId   tmdbId  Tags
0       1        1     4.0  964982703              Toy Story (1995)  Adventure|Animation|Children|Comedy|Fantasy  114709    862.0    []
1       1        3     4.0  964981247       Grumpier Old Men (1995)                               Comedy|Romance  113228  15602.0    []
2       1        6     4.0  964982224                   Heat (1995)                        Action|Crime|Thriller  113277    949.0    []
3       1       47     5.0  964983815  Seven (a.k.a. Se7en) (1995)                             Mystery|Thriller  114369    807.0    []
4       1       50     5.0  964982931   Usual Suspects, The (1995)                       Crime|Mystery|Thriller  114814    629.0    []
```

Figure 1: Peek of the dataset

such that 80 percent of the reviews by every single user is used for training the model and the remaining 20 percent of the reviews to validate the correctness of our model. We achieve the above using the train test split method in scikit-learn library. The stratify attribute of the function specifies the feature based on which the

dataset is divided into testing and training.
To gain a better intuition about the movie for content based recommendation and importance of features of movies, we integrated the data set with the publicly available IMDB and TMDB dataset, leading to features such as release date, writer, director, and cast information, tag line, overview, popularity of movie, the rating and vote counts on their respective websites.
We designed a feature, called popularity in the following manner:
We can observe that more popular movies are the ones that appear



title	count	rating
Forrest Gump (1994)	254	4.183071
Shawshank Redemption, The (1994)	250	4.430000
Pulp Fiction (1994)	249	4.162651
Silence of the Lambs, The (1991)	223	4.188341
Matrix, The (1999)	216	4.185185
...	...	...
King Kong Lives (1986)	1	2.000000
Killing Me Softly (2002)	1	4.000000
Killers, The (1946)	1	4.000000
Killers (2010)	1	3.000000
À nous la liberté (Freedom for Us) (1931)	1	1.000000

Figure 2: Popularity Feature Computed

the most number of times in the dataset.
We can observe that the most frequent Genre in the dataset is

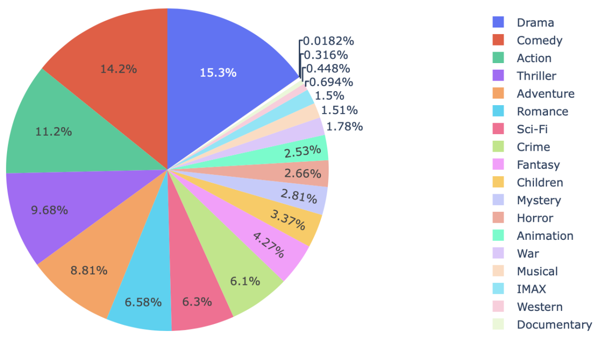


Figure 3: Distribution of Genres in the Dataset

Drama, followed by Comedy and Action.
In the dataset, we observed that the average number of user ratings per movie is 477 whereas the average number of movies rated by user = 98.
The distribution of user rating gives a better idea:
We can observe that there is a positive skew in the user ratings.

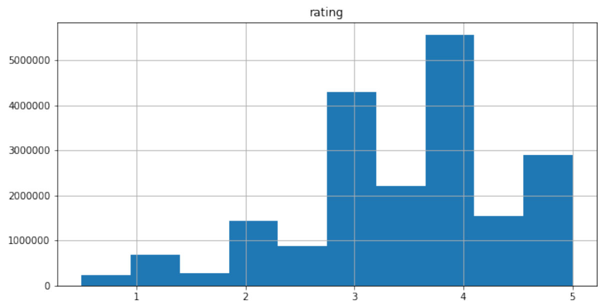


Figure 4: Histogram of user ratings

4 PROBLEM FORMALIZATION

The problem consists of two parts, first predicting the ratings and second getting the rankings or recommendations.
Metrics used for rating predictions:
Mean Absolute Error measures the average magnitude of the errors in a set of predictions, without considering their direction. The MAE is a linear score which means that all the individual differences are weighted equally in the average. It tells us how big of an error we can expect from the prediction model on average. The MAE will always be lesser than or equal to MAE. If all the errors have equal magnitude, then MAE = RMSE. The lower the value of MAE, the higher is the accuracy of the prediction model.
Formula to calculate MAE:

$$\frac{1}{n}|\sum_{i=1}^{n}(y_i - x_i)|$$

Root Mean Square Error is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data, that is, how close the observed data points are to the model's predicted values. It is a frequently used measure of the differences between values predicted by a model or an estimator and the values observed. The smaller an RMSE value, the closer the predicted and observed values are or the closer you are to finding the line of best fit. RMSE is the square root of the average of squared errors. The effect of each error on RMSE is proportional to the size of the squared error; thus larger errors have a disproportionately large effect on RMSE.
Formula to calculate RMSE:

$$(\frac{1}{n}\sum_{i=1}^{n}(y_i - x_i)^2)^{\frac{1}{2}}$$

We used the below given formulae to help us provide the top recommendations.
Metrics used for ranking predictions:

Normalized Discounted Cumulative Gain or NDCG [5] is a ranking metric which allows relevance scores in form of real numbers. It is used to evaluate the goodness of the recommendation list returned by the model. It is based on the concept that more relevant items should be placed at the beginning of the recommended list as items placed at later positions in the list tend to be overlooked by the users. Each item in the recommendation list has a relevance score which is the original rating in our case. This is called as gain

G_i. For items which don't have a relevance score (or original rating), gain is usually set to zero. To see the most relevant items at the top of the list, we divide each by a growing number which is called as discounting. We then add these discounted items to get a Discounted Cumulative Gain.

$$DCG = \sum_{i=1}^{n} \frac{rel_i}{\log_2(i+1)}$$

To make DCGs directly comparable between users, we need to normalize them. We arrange all the items in the ideal order using the actual ratings and compute DCG for them which is called as Ideal DCG (IDCG). We then divide the raw DCG by this ideal DCG to get NDCG@K, a number between 0 and 1.

$$NDCG = \frac{DCG}{IDCG}$$

Precision is calculated as the set of k items which are recommended to each user is known as top-k set. Precision at k is the proportion of recommended items in the top-k set that are relevant, that is, it is the fraction of relevant items in the top-k set. It measures the ability of the prediction model to make relevant predictions.
Formula to calculate Precision:

$$\frac{|RecommendedRelevantItems|}{|TotalRecommendedItems|}$$

Recall at k is the proportion of relevant items found in the set of top-k recommendations.
Formula to calculate Recall:

$$\frac{|RecommendedRelevantItems|}{|TotalRelevantItems|}$$

F-score is a measure of the prediction model's accuracy. It is the harmonic mean of the precision and recall.
Formula to calculate F-score:

$$F-Score = \frac{2 * precision * recall}{precision + recall}$$

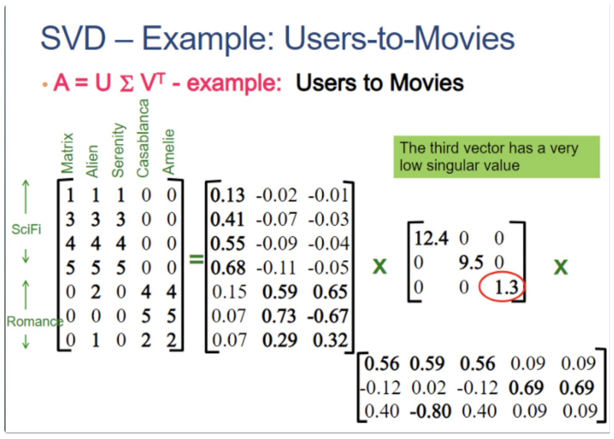
5 THE PROPOSED MODEL

We use multiple collaborative filtering approaches to build our final recommendation system. Collaborative filtering approach builds a model from a user's past behaviors (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in.

Collaborative filtering uses the ratings to form a neighbourhood N of a user, say X, whose ratings (likes and dislikes) are similar to X's ratings. The neighborhood of X is defined using nearest neighbor approach with a notion of similarity defined. There can be two types of collaborative filtering user-user and item-item. This method is different from content based as the neighborhood or similarity is defined only on the basis of rating behaviour. We have used the Surprise library prediction algorithms to analyse the performance of KNN (K-nearest neighbor) algorithm for collaborative filtering. The following are the experiments done for collaborative filtering algorithm.

1. **KNN:** For a baseline model we experimented with different KNN models.
1.1. **KNNBasic:** A basic collaborative filtering algorithm which takes the maximum number of neighbors k to take into account and the similarity metric as parameters.
1.2. **KNNwithMeans:** A basic collaborative filtering algorithm similar to KNNBasic which takes into account the mean ratings of each user.
1.3. **KNNWithZScore:** A basic collaborative filtering algorithm similar to KNNBasic which takes into account the z-score normalization of each user.

2. **SVD:** We then moved to the SVD algorithm which is based on matrix factorization [4]. SVD takes the lower dimensional representation of movies such that people who like similar movies together are mapped together. It discovers the features (hidden latent factors) such that movies can be mapped into the same space as user. We can use the SVD model to form an optimization problem to minimize the RMSE for unseen test data. Thus, using gradient descent algorithm and regularization allows rich model structure. We have used the Surprise library to analyse the matrix factorization models. We used GridCVSearch to finetune our model the to give the lowest RMSE.



SVD – Example: Users-to-Movies

• $A = U \Sigma V^T$ - example: Users to Movies

The third vector has a very low singular value

$$\begin{array}{c|ccccc} & \text{Matrix} & \text{Alien} & \text{Serenity} & \text{Casablanca} & \text{Amelie} \\ \hline \text{SciFi} & 1 & 1 & 1 & 0 & 0 \\ & 3 & 3 & 3 & 0 & 0 \\ & 4 & 4 & 4 & 0 & 0 \\ & 5 & 5 & 5 & 0 & 0 \\ \text{Romance} & 0 & 2 & 0 & 4 & 4 \\ & 0 & 0 & 0 & 5 & 5 \\ & 0 & 1 & 0 & 2 & 2 \end{array} = \begin{bmatrix} 0.13 & -0.02 & -0.01 \\ 0.41 & -0.07 & -0.03 \\ 0.55 & -0.09 & -0.04 \\ 0.68 & -0.11 & -0.05 \\ 0.15 & 0.59 & 0.65 \\ 0.07 & 0.73 & -0.67 \\ 0.07 & 0.29 & 0.32 \end{bmatrix} \times \begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix} \times \begin{bmatrix} 0.56 & 0.59 & 0.56 & 0.09 & 0.09 \\ -0.12 & 0.02 & -0.12 & 0.69 & 0.69 \\ 0.40 & -0.80 & 0.40 & 0.09 & 0.09 \end{bmatrix}$$

Figure 5: SVD in movie recommendation

3. **SAR:** The SVD approach gives us promising results but they don't really take into account the timestamp of when the user rated a movie. This feature is really important because user preferences evolve over time so our model should be able to adapt and give recent user-movie interactions more importance than old reviews.[2] So to incorporate that and explore a more advanced algorithm we implemented the Simple Algorithm for Recommendation (SAR). It is a fast scalable adaptive algorithm for personalized recommendations based on user transaction history. The core idea behind SAR is to recommend movies to users that are similar to movies that a user already has demonstrated an affinity to.

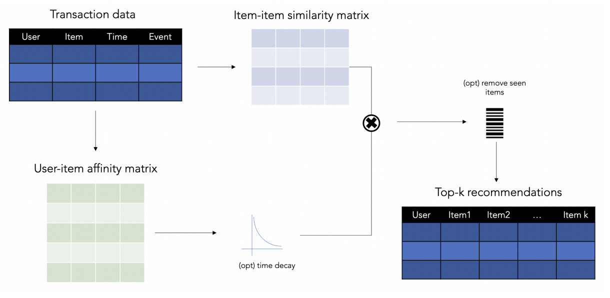


Figure 6: Simple Algorithm for Recommendation (SAR)

In the figure above we can observe that there are 2 intermediate matrix generated. Firstly S is calculated which is an item-item similarity matrix. Then a matrix A is created which stores the user-item affinity. After these the time decay is incorporated which works based on the timestamp of the user rating. This decay reduces the impact of older ratings in the recommendation. Then we multiply A and S to get the results/rating predictions. Using these predictions we give the top recommendations.

4. **GBRT (Hybrid):** To integrate the results obtained from multiple weak-learners, we decided to combine predictions of all the above models into one single hybrid recommendation system via a Gradient Boosted Regression Trees. We trained the GBRT on 3 distinct features namely rating prediction from SAR model, rating prediction from SVD model and the popularity of movies; a feature engineered by us. The figure below shows how the GBRT model takes all the predictions from difference CF models and predicts the movie ratings based on these. The hybrid model outperformed all the individual models mentioned above in terms of the evaluation metrics defined above.

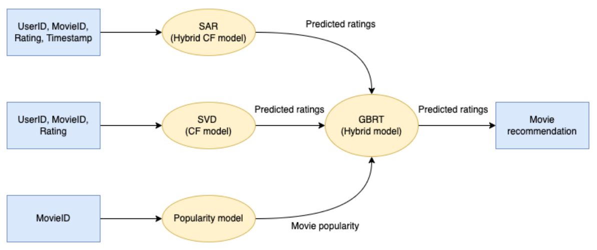
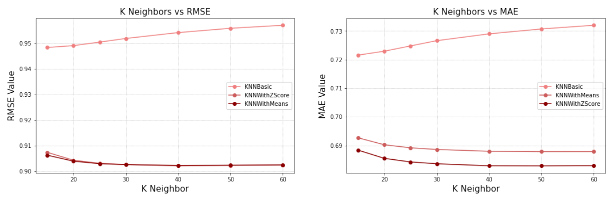


Figure 7: Pipeline for Hybrid Model

6 EXPERIMENTS

We perform the train-test split as described above 3.
1. **KNN:** We perform a GridSearchCV for different KNN algorithms based on the number of neighbors with 60 giving us the best results as seen in the figure below.



2. **SVD:** We did a GridSearchCV for SVD as well to get the optimal parameters. This gave us ’n epochs’: 25, ’lr all’: 0.01, ’reg all’: 0.4 for the best RMSE loss. The model trained with these parameters gave the following results for $top_k = 10$.

RMSE	MAE	Precision	Recall	F1 Score	NDCG
0.974338	0.874289	0.391988	0.341991	0.3652866	0.634840

3. **SAR:** We tested the SAR algorithm with different combinations. There are 3 different types of similarity metrics for user-user similarity called Jaccard, Lift and Counts. Here c_{ij} is the co-occurrence of items i and j. And s is the similarity for these two items. We get the best results using Jaccard similarity and setting the time decay option as True.

$$\text{- Jaccard : } s_{ij} = \frac{c_{ij}}{(c_{ii}+c_{jj}-c_{ij})}$$
$$\text{- lift : } s_{ij} = \frac{c_{ij}}{(c_{ii}\times c_{jj})}$$
$$\text{- counts : } s_{ij} = c_{ij}$$

These results are for top 10 recommendations for each user.

RMSE	MAE	Precision	Recall	F1 Score	NDCG
1.508195	1.214528	0.272295	0.221513	0.2442928	0.356191

4. **GBRT:** We tested the GBRT algorithm with different combinations using GridSearchCV. We got the best results with ’n estimators’: 100, ’learning rate’: 0.05, ’max depth’: 5. These results are for top 10 recommendations for each user.

RMSE	MAE	Precision	Recall	F1 Score	NDCG
0.865750	0.663193	0.491967	0.441039	0.465113	0.735813

Based on the feature importance of our model we observed that GBRT gives 85% importance to the SVD prediction parameter and about 10% to SAR predictions and 5% importance for the popularity feature.

7 CONCLUSIONS AND FUTURE WORK

We were able to build a sophisticated hybrid recommendation system based on collaborative filtering which makes use of only the user ratings. Our recommendation system takes into account the history of the user i.e. when did the user rate each movie and as the ratings become old they have less importance in the predictions. We also account for the popularity of each movie in our dataset and see that it does have a significant affect in our hybrid model’s output. Even though these recommendations show promising results we haven’t really considered the metadata of our movies. We could build a content based model which takes into account the genre and movie descriptions for predicting the movie ratings. The predictions of this content based model could be another feature in our hybrid GBRT model giving it a more diverse input set i.e. content based, popularity based and collaborative filtering based movie rating predictions to work with.

ACKNOWLEDGEMENT

We are immensely grateful to Yongfeng Zhang (Professor) and Zhiqing Hong(Teaching Assistant) for their comments on an earlier version of the manuscript, although any errors are our own and should not tarnish the reputations of these esteemed persons.

REFERENCES

[1] Sara Gasmi, Tahar Bouhadada, and Abdelmadjid Benmachiche. 2020. Survey on Recommendation Systems. In *Proceedings of the 10th International Conference on Information Systems and Technologies (ICIST '20)*. Association for Computing Machinery, New York, NY, USA, Article 10, 7 pages. DOI: http://dx.doi.org/10.1145/3447568.3448518

[2] Scott Graham, Jun-Ki Min, and Tao Wu. 2020. Microsoft Recommenders: Tools to Accelerate Developing Recommender Systems. *CoRR* abs/2008.13528 (2020). arXiv:2008.13528 https://arxiv.org/abs/2008.13528

[3] Ben Schafer, Ben J, Dan Frankowski, Dan, Herlocker, Jon, Shilad, and Shilad Sen. 2007. Collaborative Filtering Recommender Systems.

[4] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk. 2008. Investigation of Various Matrix Factorization Methods for Large Recommender Systems. *Proc. ICDM* (2008).

[5] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Tie-Yan Liu, and Wei Chen. 2013. A Theoretical Analysis of NDCG Type Ranking Measures. *CoRR* abs/1304.6480 (2013). arXiv:1304.6480 http://arxiv.org/abs/1304.6480