Comprehensive study on Movie Recommendation Systems

Group -9

Sunayana Gupta
Arizona State University
sgupt279@asu.edu

Aakash Mathai
Arizona State University
amathai1@asu.edu

Abhinav Venepally
Arizona State University
avenepal@asu.edu

Vishnuraj Kunnathveetil
Arizona State University
vkunnath@asu.edu

Dharmit Prajapati
Arizona State University
dprajap3@asu.edu

Jayashree Natarajan
Arizona State University
jnatara2@asu.edu

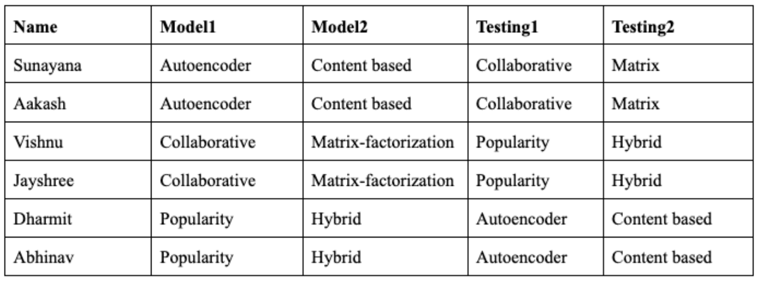
Abstract

Nowadays for taking any decision we come across a plethora of options and to decide which option choose becomes a big problem. To solve this problem we have recommendation systems but all the models come with their shortcomings hence we decided to make a model which combines the good features of all the better performing models and gives recommendations which are suitable to the user. We developed a movie recommendation system based on 'MovieLens' and TDMB dataset. Our hybrid system overcomes the shortcomings of traditional models.

1 Introduction

With the rapid increase of multimedia technologies in our society, the number of movies and videos available on social media and over-the-top (OTT) platforms is increasing, making it difficult for consumers to choose which ones to watch. Movie recommendation algorithms are extensively utilized for this purpose. For marketing purposes, movie rental outlets (e.g. NetflixTM), online shopping sites (e.g. AmazonTM), music streaming services (e.g. SpotifyTM), and other e-commerce business domains have relied heavily on tracking customers' movements, analyzing their preferences, and recommending appropriate merchandise to them. Whether we're buying a book on Amazon, watching a movie on Netflix, or reading an essay on Medium, recommender algorithms have shaped our online choices.

RS uses the information available about the user's prior behavior and preferences to identify the user's like and then makes suggestions based on that interest [1]. There are just two types of RSs in conventional techniques. The first is known as Content-based (CB) recommendation, while the second is known as Collaborative Filtering (CF). CB suggests things based on a comparison of the items' content and a user profile; CF, on the other hand, is based on user and/or item similarity measurements [7]. CF is one of the most extensively utilized and effective technologies in customized recommendation systems, as we are all aware. Goldberg et al. [3] proposed the CF method for the first time. The basic notion is that users' previous preferred behaviors have a major impact on their future behavior, and that their previous behavior is mostly consistent with their future behavior [11]. Most of the CF techniques are engaged with the users in Xu and Zhang [6] to finish the procedure with no alterations. However, consumer tastes are always evolving, and as a result, item popularity fluctuates over time. As a result, the focus of this research was on the dynamic regularization methods of the given information and data. Guia et al. [8] concentrated on the hybrid recommendation model, which



Name	Model1	Model2	Testing1	Testing2
Sunayana	Autoencoder	Content based	Collaborative	Matrix
Aakash	Autoencoder	Content based	Collaborative	Matrix
Vishnu	Collaborative	Matrix-factorization	Popularity	Hybrid
Jayshree	Collaborative	Matrix-factorization	Popularity	Hybrid
Dharmit	Popularity	Hybrid	Autoencoder	Content based
Abhinav	Popularity	Hybrid	Autoencoder	Content based

Figure 1: Figure showing distribution of work among our teammates

emphasized the availability of the user's social data, reviews, and ratings. This recommendation model is made up of six steps: review transformation, feature creation, community prediction, model training, feature blending, prediction, and, finally, assessment criteria. CF faces the issues of cold start and data sparsity as all the users rate only a very few movies and rest of the dataframe has sparse data. To overcome these probles other data is being used. the social relations between the users helps solve these problems. Several previous studies have effectively used side information to improve matrix factorization, for example [9, 10]. Due to the sparse nature of the ratings and the side information, these approaches simply use the side information as model regularizations, and the learnt latent variables may not be particularly useful. It is extremely desired to learn and extract discriminative features from datasets in order to make matrix factorization-based algorithms successful in such a situation. Deep architectures may now be used to learn hidden variables thanks to the availability of large-scale data and rich-side information. Deep learning algorithms such as Restricted Botzmann Machines, Convolutional Neural Networks, and Deep Belief Networks have been modified specifically for the job of CF [9].

Motivated by previous work we compared traditional approaches like CF, CB, SVD with the new approaches like auto encoder(AE) and hybrid model for movie recommendation algorithms in this report and how they impact our video watching experience. CF model, CB model, popularity model, AE model, and hybrid model are five types of recommender system technologies that we investigated. Furthermore, because accuracy and recall perform better in the assessment of the KNN approach [2] [5], we will utilize them to compare the results of other strategies. Since we were 6 members we had the resources required to do this comprehensive study of 6 models. To utiilize the time well we decided to split ourselves into groups of 2 each to implement and work on the models. Following is the table showing the task split. This helped us in managing the tasks and gauging the progress easily. Figure 1 shows the individual contribution of our team members on the project:

2 Dataset and data preprocessing

2.1 Data

The Movie Lens dataset was utilized in the experimental section [4] to guarantee that the suggested technique delivers the best accuracy in the recommendation system. There are two components of the dataset utilized in the experiments. The dataset contains 100,000 ratings of 9,684 movies from 643 people, as well as the relevant personal information such as age, gender, and employment. The rating distribution is shown in figure 2. This dataset was integrated with e TDMB data set to get ore features like overview of the movie and taglines given by the user.

2.2 Data Processing

We divided the dataset into test and train categories based on UserID, such that 80 percent of each user's reviews are utilized to train the model and the remaining 20% are used to evaluate the model's accuracy. The train test split function in the scikit-learn package is used to do the aforementioned. The function's stratify property defines the characteristic that divides the dataset into testing and training. We combined the data set with the publicly available IMDB and TMDB datasets to gain a better understanding of the movie for CB recommendation and the importance of movie features,

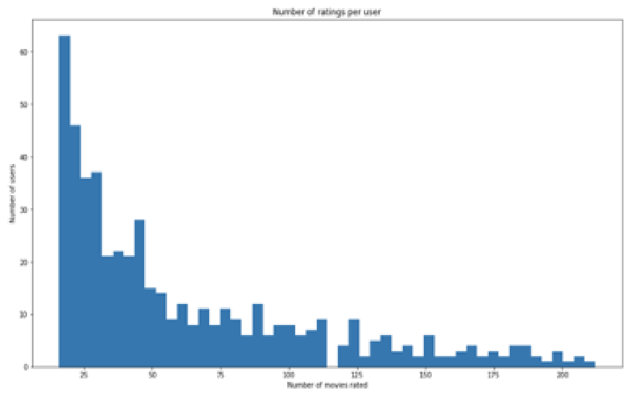


Figure 2: Figure showing distribution of ratings of movies

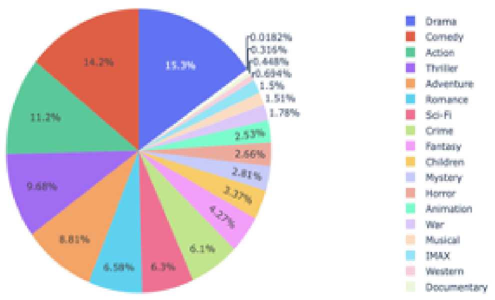


Figure 3: Figure showing distribution of movie genres

resulting in features such as release date, writer, director, and cast information, tag line, overview, and movie popularity and vote counts.

The dataset was cleaned in part by deleting duplicated and non-contributing characteristics. Genres and other categorization traits are converted to a single vector. In addition, the data must be converted into a matrix for the AE model to work. We have discussed the configuration of the matrix in detail in subsequent sections. The genre distribution of the dataset is shown in figure 3.

3 Methodology Used

3.1 Collaborative Filtering

CF is a recommendation technique that can reduce billions of items into thousands or hundreds of items[9]. It uses similarity between items and queries to make recommendations. In movie recommendation, say user A is similar to user B. If user B likes movie M, then the CF model will recommend movie M to A (even if A hasn't seen anything similar to M). There are two main benefits to using CF. It allows to make recommendations that go beyond the current interests of the user by looking at the interests of similar users. The features of items and queries are learned and not hand engineered.

In user-item filtering, the distance between items is calculated using user ratings (or likes, or whatever metric applies). When producing recommendations for a given user, we look at the user who is the most similar to the chosen user and then suggest items that similar users like but that the chosen person hasn't seen. So, if a person has seen and appreciated a certain number of films, we look at other users who have enjoyed the same movie and recommend one that haven't seen by the user yet. The item-item CF infers the relationship between distinct items based on which items are purchased together. To determine their likeness, the distance between items is determined.

For a movie recommendation system, each user's rating of the film are used to construct a vector for each user, which are then be compared using distance measures such as Euclidean distance, Cosine distance etc. To examine the performance of the KNN (K-nearest neighbor) algorithm for CF, we utilized the Surprise library prediction algorithms. The tests for the CF algorithm are listed below.

- KNNBasic: A simple CF algorithm that takes into account the maximum number of neighbors k and the similarity metric as parameters.
- KNNwithMeans: A simple CF algorithm similar to KNNBasic that considers each user's mean ratings.
- KNNWithZScore: A basic CF algorithm that takes into account each user's z-score normalization, similar to KNNBasic.
- KNNBaseline: A CF algorithm which takes into account a baseline rating.

We have used Cosine similarity to calculate the similarity between users and users or item and item. We have compared it with the Pearson correlation which is nothing but mean-centred cosine similarity.

3.2 Matrix Factorization Methods

Our recommendation model should be able to forecast the unknown ratings. We utilize the SVD latent factor approach for this. SVD uses a lower-dimensional representation of movies to map people who enjoy similar films together [10]. It finds hidden latent characteristics that allow movies to be mapped into the same space as the user. The SVD model may be used to create an optimization problem to reduce the RMSE for unknown test data. As a result of the gradient descent technique and regularization, a rich model structure is possible. We may utilize a baseline predictor to represent biases and interactions by incorporating the user bias b_x, the item or movie bias b_i, and the overall mean rating μ into its rating estimation. The matrix factorization models were investigated using the Surprise library. We have also worked with SVD_{pp} model which also takes into account the implicit ratings. The fact that a user u rated an item j, independent of the rating value, is referred to as an implicit rating. The best results are obtained by combining implicit and explicit assessments. SVDpp, on the other hand, takes the longest to train and fit and is significantly slower than other models.

The hyperparameters with the lowest RMSE were found using GridCVSearch from sklearn. Hyperparameters that work well are:

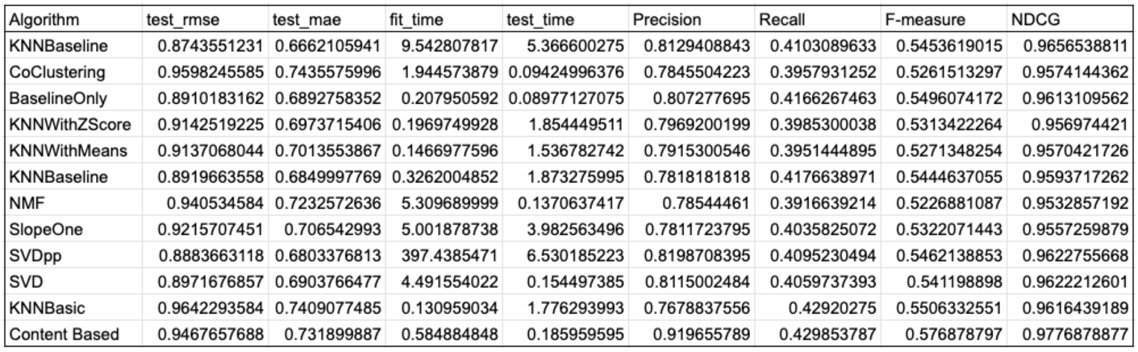
- Number of epochs (n_epochs): 10
- Number of latent factors ($n_factors$): 50
- Regularization parameter (λ) : 0.02
- Learning rate for all parameters (lr all): 0.01

3.3 Content Based Recommendation

A CB reccommender system seeks to estimate a user's characteristics or behavior based on the features of the item to which he or she responds positively. The purpose of CB model is to categorize products using particular keywords, learn what the client likes, search up those phrases in the database, and then suggest comparable items. The features on the basis of which we decided similarity between items are genre. release year, tagline and overview. Here except for genre all of the other features are taken from the TDMB dataset. We made two vectors, movie vector and user vector. Movie vector had one in the corresponding genre is that movie belongs to that genre otherwise 0 and user vector has average rating of that genre for that user.

We retrieved movie summary and taglines information by integrating MovieLens data with TMDB data. This was done using the links.csv file, we merged the dataframes using TMDB id and movie id. After that, the TF-IDF vectorizer was used to build a TF-IDF vector for each movie. Term Frequency Inverse Document Frequency is abbreviated as TF-IDF. This is a typical approach for converting text into a meaningful numerical representation, which is then used to fit a machine learning algorithm for prediction.

Its counterpart, Count tf-idf considers overall texts of weight of words, whereas vectorizer delivers number of frequency with relation to vocabulary index. In TD-IDF the highly frequent words like



Algorithm	test_rmse	test_mae	fit_time	test_time	Precision	Recall	F-measure	NDCG
KNNBaseline	0.8743551231	0.6662105941	9.542807817	5.366600275	0.8129408843	0.4103089633	0.5453619015	0.9656538811
CoClustering	0.9598245585	0.7435575996	1.944573879	0.09424996376	0.7845504223	0.3957931252	0.5261513297	0.9574144362
BaselineOnly	0.8910183162	0.6892758352	0.207950592	0.08977127075	0.807277695	0.4166267463	0.5496074172	0.9613109562
KNNWithZScore	0.9142519225	0.6973715406	0.1969749928	1.854449511	0.7969200199	0.3985300038	0.5313422264	0.956974421
KNNWithMeans	0.9137068044	0.7013553867	0.1466977596	1.536782742	0.7915300546	0.3951444895	0.5271348254	0.9570421726
KNNBaseline	0.8919663558	0.6849997769	0.3262004852	1.873275995	0.7818181818	0.4176638971	0.5444637055	0.9593717262
NMF	0.940534584	0.7232572636	5.309689999	0.1370637417	0.78544461	0.3916639214	0.5226881087	0.9532857192
SlopeOne	0.9215707451	0.706542993	5.001878738	3.982563496	0.7811723795	0.4035825072	0.5322071443	0.9557259879
SVDpp	0.8883663118	0.6803376813	397.4385471	6.530185223	0.8198708395	0.4095230494	0.5462138853	0.9622755668
SVD	0.8971676857	0.6903766477	4.491554022	0.154497385	0.8115002484	0.4059737393	0.541198898	0.9622212601
KNNBasic	0.9642293584	0.7409077485	0.130959034	1.776293993	0.7678837556	0.42920275	0.5506332551	0.9616439189
Content Based	0.9467657688	0.731899887	0.584884848	0.185959595	0.919655789	0.429853787	0.576878797	0.9776878877

Figure 4: Analysis of Collaborative filtering, Matrix Factorization and content based models

the, is are ignored as they do not convey any special meaning to the sentence. The data is then transformed into a vector space, with each vector term indexed as our index vocabulary. The term frequency counts the occurrence of each word. The inverse document frequency is calculated using below formula:

$$idf(t) = \log(\frac{|D|}{1 + |d : t \in d|})$$

where $|D|$ is the cardinality of the document space, $|d : t \in d|$ is the number of documents where the term t appears, when the term frequency function satisfies $td(t, d) \neq 0$. One is added in denominator to avoid division by 0.

Cosine similarity was determined for each movie-movie movie combination once the TF-IDF vector was generated. The cosine of the angle between two vectors projected in a multi-dimensional space is measured via cosine similarity.

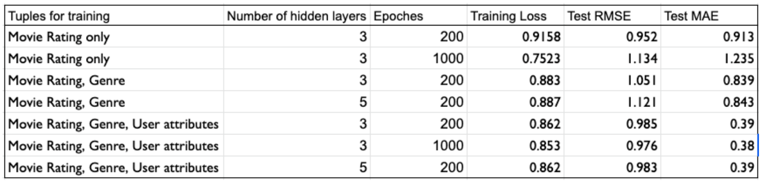
3.4 Auto Encoder Model

One of the most difficult tasks for CF and matrix factorization systems is to assign a rating when a new person or object enters the system, a situation known as the cold start scenario. The cold start problem is cyclical in that the system will not propose an item until it has some ratings, and if the system does not recommend it, it will not receive ratings. Researchers have proposed that other sources of information on users or products, sometimes known as side information, be incorporated to alleviate these issues. This secondary data might be gleaned from user/item profiles, such as a user's demographics, a movie's genre, and so on. User demographics may be utilized to deduce user relationships, and item similarity could be used to assign ratings to new things automatically. An autoencoder is a type of neural network that takes a given input and uses a deterministic mapping to map it (encode) to a hidden representation. An Autoencoder is a neural network that is meant to train an identity function in an unsupervised manner so that it may compress and reconstruct an original input, discovering a more efficient and compressed representation of the original input data in the process. There are three important blocks in AE, they are encoder $G\phi$, bottleneck z, Decoder $F\theta$. The encoder block accepts an input vector of pictures and sends them to the bottleneck as a compressed vector $'z'$, which the decoder block then attempts to reconstruct from the compressed form. The output from decoder and the input are compared using the loss calculated by:

$$L(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (x^i - f_\theta(g_\phi(x^i)))^2$$

The loss function is determined by the parameters θ and ϕ, which define the encoder and decoder, respectively. The weights and bias of the neural network are denoted by the letters G and F. So we're adding up the difference between the original picture, x, and the reconstructed image, $F(g(x))$ in the equation.

Our method combines collaborative topic regression with bayesian auto-encoders, and it necessitates the learning of a large number of hyper parameters through an EM technique. We have used multiple attributes like user age, gender registration months and extracted the genres of the all the movies to



Tuples for training	Number of hidden layers	Epoches	Training Loss	Test RMSE	Test MAE
Movie Rating only	3	200	0.9158	0.952	0.913
Movie Rating only	3	1000	0.7523	1.134	1.235
Movie Rating, Genre	3	200	0.883	1.051	0.839
Movie Rating, Genre	5	200	0.887	1.121	0.843
Movie Rating, Genre, User attributes	3	200	0.862	0.985	0.39
Movie Rating, Genre, User attributes	3	1000	0.853	0.976	0.38
Movie Rating, Genre, User attributes	5	200	0.862	0.983	0.39

Figure 5: Analysis of Auto Encoder model

predict the rating. The first n columns indicate to the number of movies that the user has rated. If the user hasn't given the film a rating, it will have a value of 0. The number of genres will be shown by the next n columns. Here, we're constructing a list of movies that the user is likely to view. As a result, the column for user 1 and Genre 1 reflects the number of movies that user 1 has rated 3 or higher in the Genre 1 category. We choose this rule to identify user preferences for a certain genre. The user's gender, age, and number of months of registration are listed in the last few rows.

We experimented with deep neural networks of three, five, and seven layers. we picked three levels since the number of layers rises exponentially with the number of layers. In our scenario, the results of 5 and 7 layered neural networks are just marginally better than the results of 3 layered neural networks. With RMS prop optimizer and sigmoid activation function, we used Mean Squared Error loss function. We can say that using movie ratings, genre counts, and user attributes with a 3 layer deep neural network and 200 epochs yielded the best results.

3.5 Popularity Model

The popularity model utilizes the popularity feature in TMDb and the weighted rating in IMDb to return the most popular movies for a given genre, We used the collective activity of the users such as average rating, number of views, likes, favorites, watchlist additions, and release date to generate the popularity feature. The collective activity mentioned above can be obtained by merging the MovieLens data with TMDB data. Using this information and the preferred preferences of the user, we recommend a list of genre-wise popular movies.

We have also considered the weighted ratings which can be calculated as:

$$W = \frac{Rv}{v+m} + \frac{Cm}{v+m}$$

where,W = Weighted Rating, R = Average rating of a movie (scale: 1-10), v = number of votes for the movie, m = minimum votes required to be listed in top and C = Mean vote average

4 Proposed hybrid model

In terms of testing MAE, RMSE, and recommendation assessment, Figure 4 and 5 show that KNNBaseline (with pearson baseline as similarity), SVDpp, BaselineOnly (takes the baseline predictor means of global ratings, user, and item/movie) and AE with 3 layers provide the best results. We used a linear combination of the ratings from the following approaches to improve the test accuracy results and balance the limits of each method. KNNBaseline aids in the introduction of the item similarity concept to users, whereas SVDpp introduces implicit ratings as well as the latent component model. The overestimations of SVDpp are balanced by SVD. BaselineOnly aids in the introduction of the user's and objects' global biases. AE adds the effect of user information as well.

We used a linear combination of the ratings from the following approaches to improve the test accuracy results and balance the limits of each method. KNNBaseline aids in the introduction of the item similarity concept to users, whereas SVDpp introduces implicit ratings as well as the latent component model. The overestimations of SVDpp are balanced by SVD. BaselineOnly aids in the introduction of the user's and objects' global biases. AE adds the effect of user information.

The hybrid model predicts ratings using a combination of the best-performing models so far, as well as the user's preference for a genre and popularity model. Given a user, we conduct the following stages, as illustrated in the diagram:

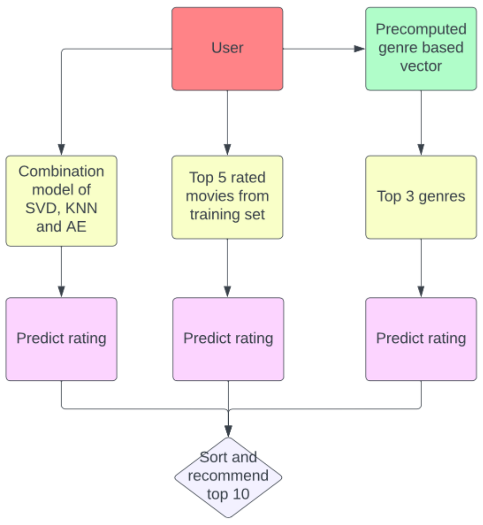
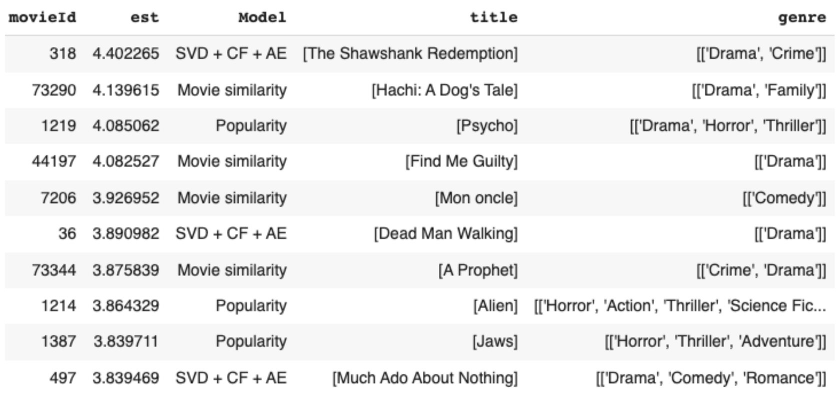

User
Precomputed genre based vector
Combination model of SVD, KNN and AE
Top 5 rated movies from training set
Top 3 genres
Predict rating
Predict rating
Predict rating
Sort and recommend top 10

Figure 6: Hybrid model



movieId	est	Model	title	genre
318	4.402265	SVD + CF + AE	[The Shawshank Redemption]	[['Drama', 'Crime']]
73290	4.139615	Movie similarity	[Hachi: A Dog's Tale]	[['Drama', 'Family']]
1219	4.085062	Popularity	[Psycho]	[['Drama', 'Horror', 'Thriller']]
44197	4.082527	Movie similarity	[Find Me Guilty]	[['Drama']]
7206	3.926952	Movie similarity	[Mon oncle]	[['Comedy']]
36	3.890982	SVD + CF + AE	[Dead Man Walking]	[['Drama']]
73344	3.875839	Movie similarity	[A Prophet]	[['Crime', 'Drama']]
1214	3.864329	Popularity	[Alien]	[['Horror', 'Action', 'Thriller', 'Science Fic...
1387	3.839711	Popularity	[Jaws]	[['Horror', 'Thriller', 'Adventure']]
497	3.839469	SVD + CF + AE	[Much Ado About Nothing]	[['Drama', 'Comedy', 'Romance']]

Figure 7: Recommendations from the hybrid model

1) Combination model: Predict the ratings for all of the films in the test set at first.
2) Movie - Movie Similarity model: From the training data, we select the top 5 highly rated movies by the user and extract comparable movies.
3) Genre Popularity Model: We discover the top three genres the user like and the five most popular movies from each genre using the user vector produced with genre.

Once we receive all of the movie ratings, we sort them in decreasing order of estimated ratings and provide the user the top ten movies to watch. Figure 7 shows the recommendation we get from the hybrid model. In this case, the estimated ratings for each film have a very high value. And the test set already comprised the best films. Figure 6 shows the complete picture of the hybrid model.

5 Results

The RMSE and MAE assessment measures were used to assess the ratings predicted by our algorithm. Other measures such as Precision, Recall, F-Measure, and NDCG were also utilized to assess the model's suggestions. To determine the suggested and relevant items utilized in the calculation of Precision and Recall metrics, we set the rating threshold t to 3.75. Things with a projected rating more than or equal to t are recommended, whereas items with an actual rating greater than or equal to t are relevant. We used the model to propose the top 10 and top 5 movies to each user, and found that the model performed better when recommending the top 5 films.

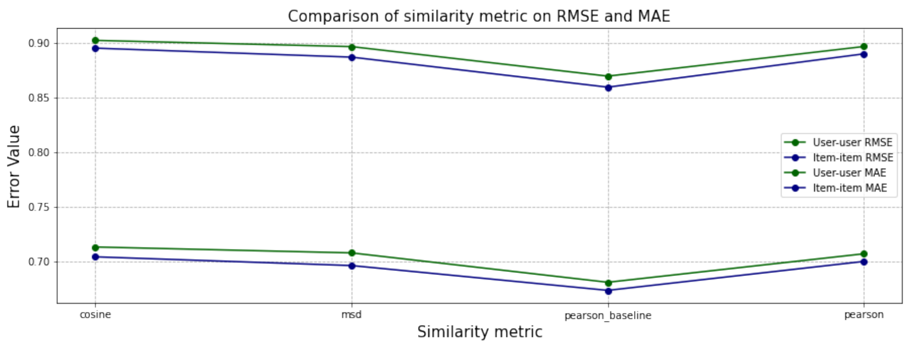


Figure 8: Comparision of similarity metrics

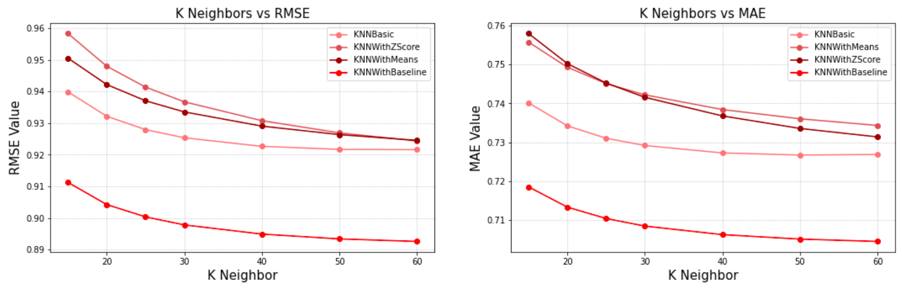


Figure 9: Comparision of similarity metrics

Among CF models, KNNBaseline produces the greatest results when it comes to creating predictions for testing data in terms of RMSE and MAE. While producing suggestions and assessing using accuracy and recall @ top-5, it also provides the greatest recall value. Figure 4 shows the results of CF, CB recommmendations and matrix factorization. When paired with the KNNBaseline method, Pearson Baseline produces the best results. As demonstrated in Figure 8, item-item based CF is more accurate than user-user based CF. Even when the user has less ratings in the training data, it performs better. Because the concept of resemblance is stronger in things than in users, the outcomes are as expected. Items, for example, can be divided into genres, with just a few overlapping categories. On the other side, users might have a wide range of tastes that can be easily measured. CF based on item-items also has the advantage of working better when a new user is added to the system (cold start problem). As we raise the maximum neighbors in the KNN algorithms, the outcomes improve as shown in figure 9.

To produce suggestion lists, we combined the above models into a single model. The combined model (SVD + CF + AE) currently recommends films from the testing set. As the final recommendation list, we collected the top movies from the popularity model, combined model (SVD + CF + AE), CB (movie similarity), and selected the top movies based on projected ratings. Though we don't have the initial ratings, we can't use conventional parameters to evaluate the model, the quality and diversity of suggestions have increased. The results increase for users who have rated less movies, demonstrating that the hybrid model is an effective solution to the common cold start problem in recommendation systems.

6 Conclusion

For producing suggestions and comprehending the concept of similarity, the CB and popularity-based models are adequate. They can help a new user overcome the difficulty of a cold start. However, they lack customisation for a specific user, which the AE model successfully addresses. CF and latent factor approaches are more powerful for big datasets and capture user-related information well. As observed in the combined model, combining the CF, SVD, and AE models enhances the accuracy of anticipated ratings. The hybrid model combines the best characteristics of both models to provide a balanced list of recommendations for a user.

Acknowledgments

First and foremost, we would like to express our gratitude to our Professor, Dr. Kookjin Lee, for providing us with the opportunity to work on this project, which allowed us to obtain a thorough understanding of recommendation systems. Next, we'd like to express our gratitude to our teaching assistants, Kavya Bathini and Niveditha Natarajan for assisting us throughout the course. We'd also like to express our gratitude to our colleagues for their unwavering support throughout the project.

A Appendix

Rating evaluation parameters

A.1 Root Mean Square Error (RMSE)

The square root of the variance of the residuals is the root mean square error. It shows how well the model fits the data in terms of absolute fit, or how closely the observed data points match the model's anticipated values. It's a commonly used metric for comparing the values predicted by a model or estimate to the values observed. The closer the anticipated and observed values are, or the closer you are to finding the line of best fit, the lower the RMSE number. The square root of the average of squared errors is the RMSE. The impact of each error on RMSE is proportional to the squared error size, thus greater errors have a disproportionately significant impact on RMSE.

Formula to calculate RMSE:

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

where, n is the number of data points, y_i is the predicted value, x_i is the actual/observed value.

A.2 Mean Absolute Error

The Mean Absolute Error (MAE) is a statistic that assesses the average size of mistakes in a set of predictions without taking their direction into account. The MAE is a linear score, which implies that the average weights all individual differences equally. It indicates the average inaccuracy that the prediction model will produce. MAE will always be smaller or equal to MAE. MAE = RMSE if all errors are of the same magnitude. The prediction model's accuracy increases as MAE decreases.

MAE is calculated by using the following formula:

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

n is the number of data points, y_i is the predicted value, x_i is the actual/observed value.

Recommendation Evaluation parameters

A.3 Precision

The top-k set is a collection of k things that are suggested to each user. Precision at k is the percentage of relevant things in the top-k set, or the proportion of suggested items in the top-k set that are relevant. It assesses the prediction model's ability to create accurate predictions. For example, if a top-5 recommendation model's Precision at 5 is estimated to be 80%, it signifies that 80% of the suggestions given by this model are relevant to the user. As a result, we strive to improve the correctness of our model at all times. Precision calculation formula:

$$Precision = |RecommendedItemsthatarerelevant|/|RecommendedItems|$$

where (with a threshold rating value of t), Recommended Items that are relevant are items with both the original and predicted ratings greater than or equal to t, and Recommended items are items with predicted rating greater than or equal to t.

A.4 Recall

The proportion of relevant items identified in the collection of top-k suggestions is known as recall at k. For example, if our prediction model's recall at 10 is found to be 35%, this indicates that 35% of the total number of relevant items appear in the top-10 suggestions. As a result, we strive to maximize the model's recall value. Recall Calculation Formula:

$$Recall = |RecommendedItemsthatarerelevant|/|RelevantItems|$$

where while (with a threshold rating value of t), Items having both the original and anticipated ratings greater than or equal to t are recommended as relevant. Items with an initial rating greater than or equal to t are considered relevant.

A.5 NDCG

NDCG is a ranking measure that allows for real-number relevance ratings. It is used to assess the model's suggestion list's quality. It is founded on the idea that more relevant things should be at the front of the suggested list, as those near the bottom of the list are more likely to be neglected by users. Each suggestion item has a relevance score, which in our instance is the original rating. Gain Gi is the term for this. Gain is commonly set to zero for things that don't have a relevance score (or initial rating). We divide each by an increasing number (typically a logarithm of the item position) to display the most relevant things at the top of the list, which is known as discounting. The discounted products are then added to create a DCG (Discounted Cumulative Gain).

$$DCG = \sum_{i=1}^{n} rel_i / \log(i+1)$$

We must standardize DCGs in order to make them directly comparable between users. Using the actual ratings, we arrange all of the items in the ideal order and compute DCG for them, which is referred to as Ideal DCG (IDCG). The raw DCG is then divided by the ideal DCG to obtain NDCG@K, a value between 0 and 1.

$$NDCG = DCG/IDCG$$

References

[1] Bagher Rahimpour Cami, Hamid Hassanpour, and Hoda Mashayekhi. A content-based movie recommender system based on temporal user preferences. In *2017 3rd Iranian Conference on Intelligent Systems and Signal Processing (ICSPIS)*, pages 121–125. IEEE, 2017.

[2] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46, 2010.

[3] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[4] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[5] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296, 2011.

[6] Zhang Na and Wang Zhi-Yong. Collaborative filtering recommendation algorithm based on hybrid similarity. In *Recent Developments in Intelligent Computing, Communication and Devices*, pages 617–625. Springer, 2019.

[7] Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

[8] Marcio Guia Rodrigo, Rocha Silva, and Jorge Bernardino. A hybrid ontology-based recommendation system in e-commerce. *Algorithms*, 12(11), 2019.

[9] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.

[10] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658, 2008.

[11] Andreas Töscher, Michael Jahrer, and Robert Legenstein. Improved neighborhood-based algorithms for large-scale recommender systems. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–6, 2008.