Hybrid Model for Movie Recommendation

Bhuvan M S
msbhuvan@uw.edu

Darshan Mehta
darshanm@uw.edu

Yash Kale
yashkale@uw.edu

1 Introduction

Movie recommendation systems are widely used by websites such as IMDB, TMDB, and Netflix. A few of the many tasks of such systems include predicting accurate ratings, solving the cold-start problems for users, and addressing popularity bias. While most implementations employ a combination of the ratings matrix and the movie metadata, there has been little emphasis on movie recommendation based on the movie poster or the synopsis of the movie plot. We hypothesize that the probability of a user watching a specific movie is a complex function of various features such as the preference of the user for similar movies (captured by the ratings of the user to those movies), ratings of similar users for the movie, various content-based features of the movie like the run-time, language, cast and also the textual summary of the plot and the features of the movie poster. Therefore, in this project, we explore a hybrid recommendation system which models each of the above-mentioned aspects of the movie to predict the ratings for a user-movie pair.

We build a hybrid model as an ensemble of various methods like Item-Item-based Collaborative Filtering, User-User-based Collaborative Filtering, Latent Factor Model, Metadata-Content-based Model, Plot-Content-based model, and Poster-Content-based model. We plan to explore how to combine the predictions from each of these models to get a final rating for a user-movie pair. We also plan to explore how useful each of these models are to address specific issues such as the cold-start problems for a new user, catering to the unique tastes of users and recommending new/unpopular movies, and recommendations for specific genres. For all of our experiments, we work with 'The Movies Dataset' [1].

2 Literature Review

Tuinhof et al. [2] propose a two-stage deep learning framework for recommending fashion products based on an input picture. They use two deep convolutional neural networks which have been pretrained on a large product database and fine-tune it for their dataset. The concatenated values from the second last layer of both these networks are then used as a feature representation of the image content. They use k-NN to search for the closest item in their dataset. While their feature extraction process seems to be astute, the k-NN model which runs on top of these features can get out of hand really quick as the data size increases. This is especially a problem when recommending in real-time. Using techniques such as Locality Sensitive Hashing (LSH) on top of the features extracted from the posters which would reduce the time complexity significantly.

Bergamaschi et al. [3] use topic models from the summary of the 'plot' associated with each movie. They use a content-based recommendation system which evaluates the similarity between the plot

Preprint. Under review.

of the movie with the plots of other movies in the dataset. Since this method is independent of user ratings, hence, it wouldn't have the popularity bias issue. We emphasize on the representation of the plot text detailed in this paper. Each plot text is preprocessed by first removing stop-words, then performing lemmatization, and finally vectorizing it using TF-IDF technique [6]. Furthermore, dimensionality reduction is performed using Latent Dirichlet Allocation (LDA) [7] and a Singular Value Decomposition (SVD) based approach called Latent Semantic Analysis (LSA) [8]. The similarity of the movies is assessed based on the cosine similarity score taken over the topic space defined by LDA or LSA. While the cost of computation of LDA matrix was lesser than LSA in terms of both time and space, the authors, however, found during the evaluation that the quality of topics and the recommendations generated using LSA were better than using LDA. Upon further research on dimensionality reduction methods, we found that Doc2vec outperformed LSA when working which large corpora as mentioned in [13] and hence we decided to use Doc2vec.

Viktoriia et al. [4] classify hybridization into 7 major categories some of which are:

- **Weighted**: implementing different methods separately and then combining their predictions using weighted average.
- **Feature Combination**: features from different recommender systems data sources are put into a single recommendation algorithm.
- **Feature Augmentation**: the output of one system is used as an input feature to another, for example, using the model generated by one to generate features that are used by another.
- **Mixed**: incorporates two or more techniques at the same time, e.g.: Content-based and Collaborative Filtering.

They compare and contrast several state-of-the-art papers in order to understand which of these hybridization techniques work better. In their findings, they claim that weighted hybridization of Content-based and Collaborative Filtering models are by far the most popular technique. Similar to the papers reviewed by the authors, we plan to try how different combinations of models improve the recommendations in different scenarios by performing an exhaustive study of these combinations.

3 Problem Statement

We focus on two major tasks in this paper:

- **Rating prediction**: We construct multiple models which predict ratings for the user-movie pairs in the test dataset.
- **Evaluation of models in different scenarios**: We study the performance of all the models and their hybrid variants on binned test data. For example, we bin the data on the basis of the number of ratings that the movies have received and then study the performance of each model on each bin. In a similar fashion, we perform binning on users based on the number of movies that they have rated. Also, we evaluate the performance of each model for the top 10 frequent genres in the dataset. This would provide us with insights on which model is better suited for which scenarios.

More details on how each model works and the challenges we would have to address are discussed in Section 5.

4 Dataset

For this study, we use 'The Movies Dataset' [1] available on Kaggle. This dataset is a merger of two different datasets: the 'MovieLens 20M' [14] which consists of the ratings matrix and the TMDB and IMDB IDs of movies and the TMDB movie metadata dataset which was scraped using the API provided by TMDB. The metadata dataset contains detailed information about each movie such as the list of cast and crew, genre, title, languages, release data, plot (synopsis) of the movie, link to the poster of the movie, etc.

We preprocess the above datasets and produce four different components as follows:

1. **Rating Matrix**: This contains the ratings of each user-movie combination in the dataset. We have a total of 270897 users and 45115 different movies out of which 26024289 ratings are available. This ratings matrix is 99.78% sparse in nature and hence we store it in a sparse matrix suitable format using a dictionary of dictionaries.
2. **Movie-Metadata Matrix**: From the metadata matrix, we pick the following columns: genre, languages spoken, the runtime of the movie, popularity of the movie, average IMDB rating, and the list of top actors in the movie. We define an actor to be a top actor if they have been in more than 50 movies. This number was picked heuristically after observing the distribution of the number of occurrences of actors in movies. We first one-hot encode all the categorical values and then use PCA to reduce the vector size to 512 dimensions.
3. **Movie-Poster Content Matrix**: We scrape the links of the movie posters available in the metadata dataset for each movie to obtain its poster. We then create an autoencoder where both, the input and output of the autoencoder, is the poster. This neural network has an intermediate layer of size 1024 which it must learn in a way such that it is able to reconstruct the original poster given just these 1024 values. The outputs of this layer of size 1024 are then used to generate the poster embeddings which is then accumulated into what shall be called the Movie-Poster Content Matrix.
4. **Movie-Plot Content Matrix**: For each movie present in the metadata dataset, we generate word embeddings for the plot of the movie. Initially, we remove the stop words from the plots, lemmatize the plot and then pass the plot to a Doc2Vec model [10] to generate a vector of size 300. Each plot is now represented by a vector of size 300 based on the words present in the plot.

5 Methodology

We propose a hybrid model to predict the ratings for each user-movie pair. Firstly, the data is randomly split and 650,000 entries of the ratings matrix are assigned to each, the validation set and the test set. We build 6 different models (on our own without using any external pre-built models) from our data and each of these models would predict a rating (real number between 0 and 5) for each user-movie pair. An ensemble model would then aggregate the ratings from each model to give a final rating for each user-movie pair.

Following are the 6 models which are being used in our method:

- **Item-Based Collaborative Filtering (CF-Item)**: It uses the ratings matrix. For a given user-item pair (x, i), the predicted rating would be the weighted average of the ratings of each of the other items j that the given user x has rated ($N(i; x)$) as shown in the equation 1, where the weights are the similarity between the given item to each other item (sim_{ij}) measured using the Pearson Correlation Coefficient [5] and the bias term b_{xi} is the sum of the overall mean movie rating (μ), rating deviation of movie i (b_i), and rating deviation of user x (b_x).

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} sim_{ij}.(r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} sim_{ij}} \tag{1}$$

- **User-Based Collaborative Filtering (CF-User)**: It uses the ratings matrix. For a given user-item pair (x, i), the predicted rating would be the weighted average of ratings from each of the other users y who have rated the item i ($N(i; x)$) as shown in the equation 2, where the weights are the similarity between the given user to each other user (sim_{xy}) measured using the Pearson Correlation Coefficient [5] and the bias term b_{xi} is the sum of the overall mean movie rating (μ), rating deviation of movie i (b_i), and rating deviation of user x (b_x).

$$r_{xi} = b_{xi} + \frac{\sum_{y \in N(i;x)} sim_{xy}.(r_{yi} - b_{yi})}{\sum_{y \in N(i;x)} sim_{xy}} \tag{2}$$

- **Latent Factor Model (LFM)**: It uses the ratings matrix. We consider $k = 20$ factors, and factorize the ratings matrix (movies vs users) into Movie-Factors (Q) and Factors-Users (P^T) matrices. We learn the matrices Q and P by minimizing the objective function (shown in equation 3) using Stochastic Gradient Descent (SGD) [13]. For a given user-item pair (x, i), the predicted rating would be the $q_i p_x^T$ where q_i and p_x are vectors corresponding to item i and user x from Q and P respectively, and $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ are the regularization coefficients (found using cross-validation), b_x is the rating deviation of user x, and b_i is the rating deviation of item i.

- **Content-Based Model - Metadata (CB_Metadata)**: It uses the standardized Movie-Metadata Content Matrix. For a given user-item pair (x, i), each i^{th} row vector of this matrix represents an item-profile ***i***, and the user-profile ***x*** is created by taking the weighted average of all the item-vectors for items rated by user ***x*** where the weights are the respective ratings. The similarity metric used for comparing the item-profile ***i*** and the user-profile ***x*** is euclidean distance.
- **Content-Based Model - Plot Text (CB_Plot)**: It uses the Movie-Plot Content Matrix, where each entry is represented as an embedding obtained using the Doc2Vec model [10]. For a given user-item pair (x, i), each i^{th} row embedding of the Movie-Plot Content Matrix represents an item-profile ***i***, and the user-profile ***x*** is created by taking the weighted average of all the item-vectors for items rated by user ***x*** where the weights are the respective ratings. The similarity metric used for comparing the item-profile ***i*** and the user-profile ***x*** is cosine similarity.
- **Content-Based Model - Poster (CB_Poster)**: It uses the Movie-Poster Content Matrix, where each entry is represented as an embedding obtained using the autoencoder. For a given user-item pair (x, i), each i^{th} row embedding of the Movie-Poster Content Matrix represents an item-profile ***i***, and the user-profile ***x*** is created by taking the weighted average of all the item-vectors for items rated by user ***x*** where the weights are the respective ratings. The similarity metric used for comparing the item-profile ***i*** and the user-profile ***x*** is euclidean distance.

$$\min_{Q,P} \sum_{(x,i)\in R} (r_{xi} - (\mu + b_x + b_i + q_i p_x))^2 + (\lambda_1 \sum_i \|q_i\|^2 + \lambda_2 \sum_x \|p_x\|^2 + \lambda_3 \sum_x \|b_x\|^2 + \lambda_4 \sum_i \|b_i\|^2) \tag{3}$$

Each of the above models is trained on the training data. In the case of Latent Factor Model, we perform cross-validation on the training data in order to estimate suitable values for the regularization coefficient. In the case of Content-based models, only the embeddings associated with the user-item pairs occurring in the training data are used to make the predictions. Once trained, each of these models is used to make predictions on the validation set. Next, an ***ensemble*** of the above models would be created as a linear combination of the predictions from each of the models where the weights would be learned by training a Ridge Regression model [12]. This training is done using the predictions made by the 6 models on the validation set. We perform cross-validation on the validation data in order to estimate suitable values for the regularization coefficient of the ensemble model.

5.1 Evaluation Metrics

We use the following evaluation metrics for both, the individual models and the ensemble models:

1. **ROC_AUC @** k (Primary): This metric measure the area under the ROC curve where we first binarize (whether a movie is recommended to a user or not) the ground truth ratings by setting k as the threshold value. Next, we divide each prediction made by the model by 5 (since ratings are on a 0-5 scale), and then compute the area under the ROC for these scaled predictions. Since we focus more on getting the higher ratings correct, this metric is quite suitable for measuring the same. The drawback of this metric is that we lose a lot of valuable information with respect to the exact real-value of the predicted rating in the process of binarization.
2. **RMSE @** k (Secondary): This metric measure the root mean squared error where we only consider the user-item pairs having a rating of at least k in the ground truth for the calculation of this value. The benefit of making a soft cut is that we get to focus on the higher ratings while not losing out on a lot of valuable information, but the drawback is that we completely disregard the predictions of the model for inputs having ground-truth rating of less than k.

5.2 Experimental Design

In order to evaluate the efficiency of the models, we design a series of experiments hoping to get a better insight into the data and the models.

5.2.1 Overall Ratings Prediction Performance Analysis

First, we plan to evaluate all the models on the entire testing set. We have 6 models (CF_Item, CF_User, LFM, CB_Plot, CB_Poster, and CB_Metadata) plus an ensemble model. We evaluate

each of these on ROC_AUC at three different thresholds: 2, 2.5, 3, and on RMSE at three different thresholds: 2, 2.5, 3. We also plot the ROC curve for each of the models to compare their performance.

5.2.2 Model Analysis Across User Segments

Via this experiment, we wish to explore if certain models are better catered than others for specific segments of users. We segment users into deciles (group of 10 consecutive percentiles) based on the number of movies they have watched. We then retrain all the 6 main models for each of the user segments. Next, we plot the ROC_AUC @ 2.5 and the RMSE @ 2.5 value of each of the models on each of the segments. This experiment could also help us identify which model would be best recommended for users who are very new to the platform and have watched very few movies.

5.2.3 Model Analysis Across Movie Segments (Rating Count)

Via this experiment, we wish to explore if certain models are better catered than others for specific segments of movies. We segment movies into deciles (group of 10 consecutive percentiles) based on the number of ratings they have received. We then retrain all the 6 main models for each of the movie segments. Next, we plot the ROC_AUC @ 2.5 and the RMSE @ 2.5 value of each of the models on each of the segments. This experiment could also help us identify which model would be best recommended for movies which have freshly been added to the platform. It would also help us identify models which are good at recommending niche movies.

5.2.4 Model Analysis Across Movie Segments (Genres)

Via this experiment, we wish to explore if certain models are better catered than others for specific genres of movies. For the purpose of this study, we pick the top 10 genres encompass have the highest number of movies. These are: Action, Adventure, Comedy, Crime, Documentary, Drama, Horror, Romance, Science Fiction, and Thriller. We segment the movies based on which genre they belong to. We then retrain all the 6 main models for each of the movie segments. Next, we plot the ROC_AUC @ 2.5 and the RMSE @ 2.5 value of each of the models on each of the segments.

5.2.5 Personalization Of Ensemble Models

In this experiment, we wish to explore if personalizing the ensemble model for each segment would help in improving the prediction performance. For this, we first train a separate ensemble model for each segment of user segments experiment and each segment of the movie segment (rating count) experiment. This ensemble would only be trained on the subset of validation data which corresponds to that segment. At test time, when a new user-movie pair is passed as a query, we redirect the query to the ensemble model corresponding to the user/movie segment the query belongs to. Finally, we compare the base ensemble model which was trained on the entire validation data to the personalized versions which were trained on the user segments and the movie segments. The metric used for comparison is ROC_AUC @ 2.5 and RMSE @ 2.5.

6 Results and Discussion

In this section, we first compare the performance of each of our models in predicting the ratings on the test set of our ratings matrix. Further, we perform various experiments to analyze the performance of each model in different scenarios to extract insights about the strengths and weakness of each of the models.

6.1 Ratings Prediction Performance Analysis

Each of the 6 individual models - CF_Item, CF_User, LFM, CB_Plot, CB_Poster and CB_Metadata, and an ensemble Ridge Regression model as developed as described in 5 section. The predictions from each of these models on the held-out test-set was used to compute the primary and secondary performance metrics, which have been summarized in the Table 1.

We observe that the ensemble model and the Latent Factor Model give the highest ROC-AUC @ 2.5 of 0.83, which is significantly higher than other models. Also, we can observe that the performance of both the Collaborative filtering models (CF-User and CF-Item) were comparable, but CF-User

Table 1: Comparison of each individual model and the ensemble model performance on the test set based on ROC-AUC and RMSE.

	AUC@2	**AUC @ 2.5**	AUC @ 3	RMSE@2	**RMSE@2.5**	RMSE@3
CB_Metadata	0.451	*0.455*	0.449	1.445	*1.269*	1.176
CB_Plot	0.531	*0.523*	0.517	1.666	*1.727*	1.774
CB_Poster	0.505	*0.5*	0.499	1.012	*0.973*	0.975
CF_Item	0.755	*0.745*	0.739	0.936	*0.893*	0.88
CF_User	0.784	*0.767*	0.762	0.827	*0.783*	0.775
LFM	0.853	*0.836*	0.835	0.715	*0.672*	0.666
Ensemble	0.854	*0.836*	0.836	0.719	*0.677*	0.671

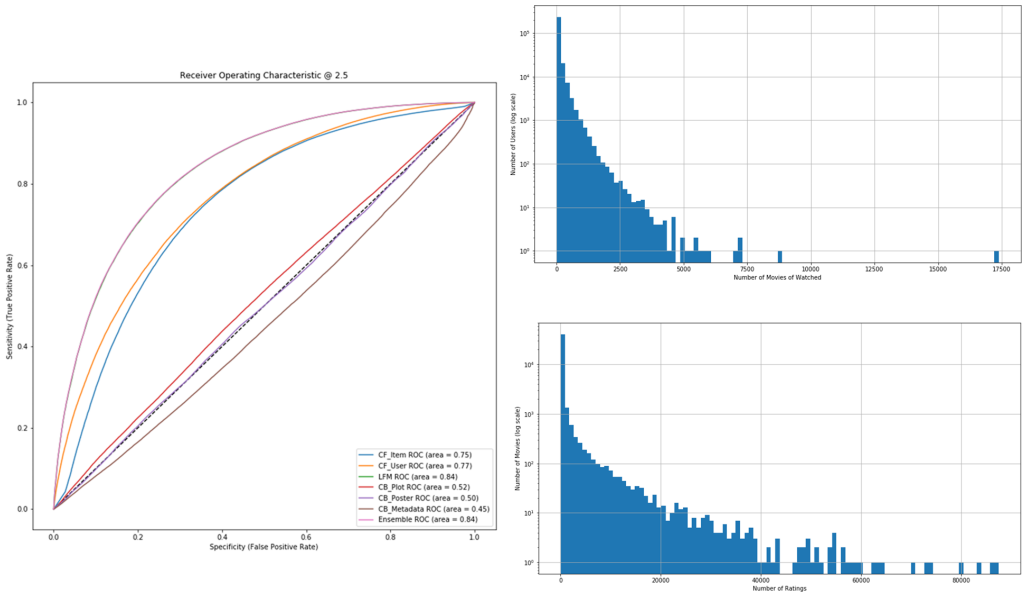

Figure 1: (Left) ROC Curve of multiple models on the test data predictions with ground-truth labels binarized at a threshold of 2.5. We can observe that the Ensemble model and the LFM work significantly better than the other models. (Right Top) The histogram of users showing a long tail distribution. (Right bottom) The histogram of movies showing a long tail of distribution, highlighting the popularity bias problem.

performed slightly better (with score of 0.767) than the CF-Item model which gave a score of 0.745. In contrast, we can see that all three Content-Based models perform very poorly on this dataset resulting in an ROC AUC @ 2.5 score close to 0.5. The ensemble model which was build by regressing over the predictions from each of the individual models also captured this trend, by learning a higher coefficient value corresponding to the predictions from LFM. The data was normalized before training the ensemble model using Ridge Regression, hence we can interpret the coefficient values as an indicator of feature importance. In order to verify the robustness of our chosen rating-threshold of 2.5 for the ROC computation, we also show the ROC-AUC scores at different thresholds. We observe that the relative performance of the models is consistent across different rating-thresholds.

Similar observations can be made on the secondary evaluation metric RMSE @ 2.5, where ensemble and LFM model had an RMSE @ 2.5 of about 0.67. The relative performance of each of the models was consistent with both the primary and secondary evaluation metrics.

Therefore, we can conclude that the LFM model is most suited for this problem as it gives comparable performance to the ensemble model, and it is easier to maintain (regular retraining for continuous improvement), as we need to train a single model. Moreover, in terms of computational efficiency, LFM consumes the least amount of space as it stores only a matrix across a small number of latent factors, and is very fast during inference time as shown in Table 2 (average of only 1.12 microseconds

Table 2: Comparison of inference time taken by CF-User, CF-Item an LFM models.

Model	Time-Taken
CF-Item	320 milli-seconds
CF-User	1.65 seconds
LFM	1.12 micro-seconds

per prediction for a user-movie pair, compared to CF-Item model which takes an average of 320 milliseconds per prediction.)

6.2 Model Analysis Across Different Segments

Each of the different recommendation system model work on different feature spaces towards the same goal of predicting an accurate ratings for a given user-movie pair. Therefore, it is crucial for us to identify the best-suited models for different scenarios.

Here, we try to analyze how our individual models perform in different scenarios by evaluating the models across different segments of users and movies. For example, model analysis of 'Popular movies vs Non popular movies' is an example where one segment refers to the dataset divided into movies which are popular (rated by many users) and other segment refers to those movies which are not popular (rated by very few users). The segments are formed based on the types of questions we ask or scenarios we want to explore.

We define three scenarios which we explore for this analysis from which we derive interesting insights about the behavior of different models.

6.2.1 User Segments:

We explore whether any of our models perform better/worse for any particular segment of users. The users are divided into deciles based on the number of movies watched by them as shown in Figure 2. The lower deciles contain users who have watched very less movies, and the higher deciles refers to the users who have watched relatively larger number of movies. The lower deciles represent the a scenario which approaches a cold-start problems, where we have very less prior knowledge about the user, hence it becomes hard to predict the ratings well for such users. In these cases, content based models might be effective if we have a good feature representation. However, in our experiments we found that the content based methods worked relatively poorly compared to other models even in the lower deciles. This could to due to the fact that our metadata features were noisy, and the feature embeddings for the unstructured data (plots and posters) might not have been very effective due to lack to supervised information. However, we observe that the LFM and CF-User and CF-Item model perform quite well at low deciles (ROC-AUC@2.5 ≈ 0.76). This could be due to the inclusion of bias terms in the equations of LFM and Collaborative Filtering methods.

We observe that the LFM and CF-User model show an average uptrend as number of movies watched per user increases. As a user watches more and more movies, we get more data which could help in getting more reasonable similarity scores, leading to an increased performance in the LFM and CF-User model at higher deciles. This is particularly noticeable as we approach the 90th percentile. There is a cross over between the CF-User and CF-Item models. Therefore, we can derive an insight that the CF-Item model performs marginally better at lower deciles of user-segments, whereas the CF-User model performs slightly better for higher deciles. Moreover, we can consider that the LFM is effective in and performs best across all user deciles.

6.2.2 Movie Segments:

We explore another set of scenarios to analyze the performance of these models for movies of different popularity level. The movies are divided into deciles based on the number of users that have rated those movies as shown in Figure 3. The lower deciles represent the set of movies which have been rated by very less number of users, whereas the higher deciles represent popular movies which have been rated by lot of users. As we can see from Figure 1 (bottom right, y-axis in log scale), that there are high number of unpopular movies, which leads to popularity bias. From 3, we can observe that the Collaborative filtering models do not perform well at lower deciles, but they perform significantly

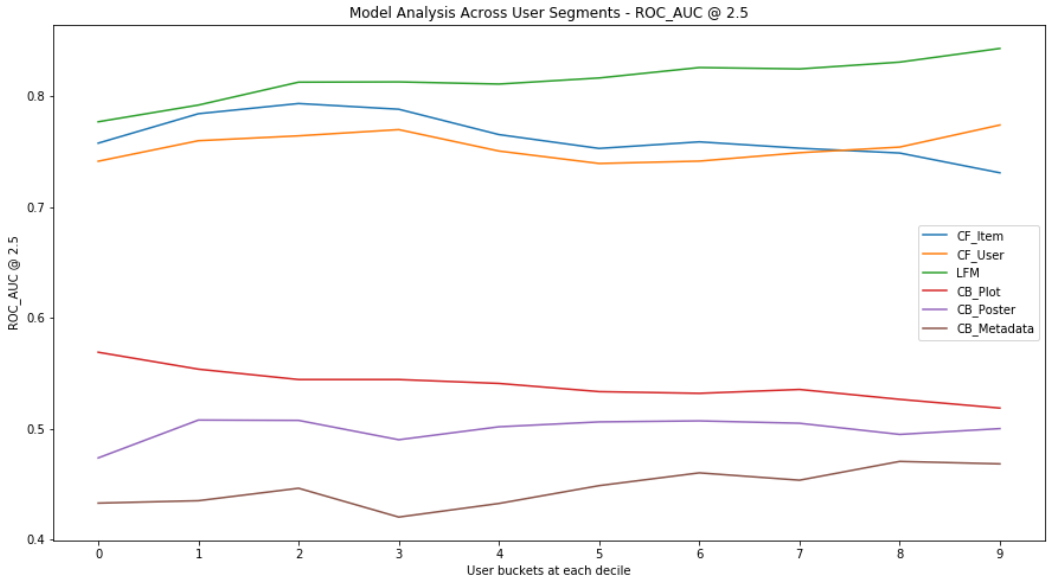

Figure 2: Comparison of ROC AUC @ 2.5 scores of different models across multiple segments (deciles) of users. We observe that most of the models (especially LFM) perform better for users who have watched more movies.

better at higher deciles. Therefore, we can derive an insight that the Collaborative Filtering models are not suited well for predicting the ratings for unpopular movies.

Also, we observe that, both the CB-Plot and the CB-Poster models initially show a high ROC value, but immediately show a sharp decline as the number of users that have watched that movie increases(i.e movie occurs in larger number of user profiles). However, the model performance of Collaborative Filtering and LFM improves as the number of users that have rated the movie increases. This might be the case because, as more users rate a movie, we get a more general understanding of the rating of the movie, leading to stabilization which we can observe in CF-User and CF-Item models at the 90th percentile.

6.2.3 Genres:

We have taken the top 10 genres (based on the frequency in the dataset) to explore how our models would fare across each genre as shown in 4. We observe that the LFM model has outperformed all the other models across all genres. We can observe that the models perform consistently across various genres.

6.3 Personalization Of Ensemble Model

The results of this experiments, as seen in Table 3, show that there was a very minute improvement in the ROC_AUC @ 2.5 and the RMSE @ 2.5 metric. The cost of implementation and maintenance of this personalization feature is very high, but the gain seems to be quite low. In practical applications, spending more resources on personalization for a minute gain would be a bad idea.

6.4 Limitations and Future Work

This section highlights few major limitations of the project and possible improvements.

We clearly observe that the Content-Based models perform poorly compared to other models. This could be due to the fact that the Plot-based feature embeddings and the Poster-based feature embeddings created without fine-tuning on any supervised input, led to poor representation of the movies. Therefore, the similarities of movies in this feature-space was probably not very helpful in predicting the ratings. Another possibility could be that the plots and posters by themselves are not a good predictors of ratings. Since content-based models have potential to work well on the cold-start prob-

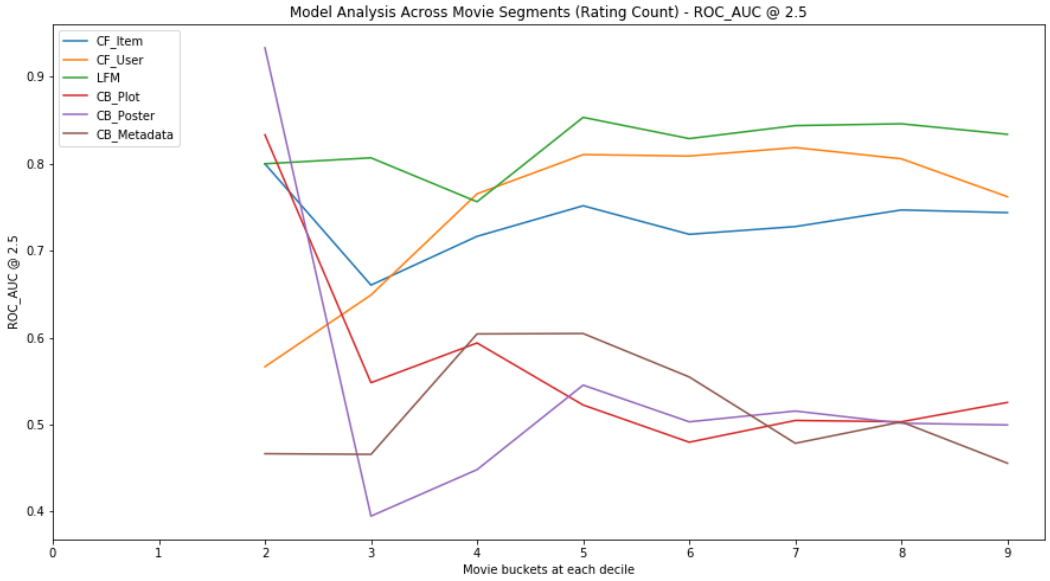

Figure 3: Comparison of ROC AUC @ 2.5 scores of different models across multiple segments (deciles) of movies. We can observe that the Content based models performed at par with other models in the long tail region where movies have been watched by very few users.

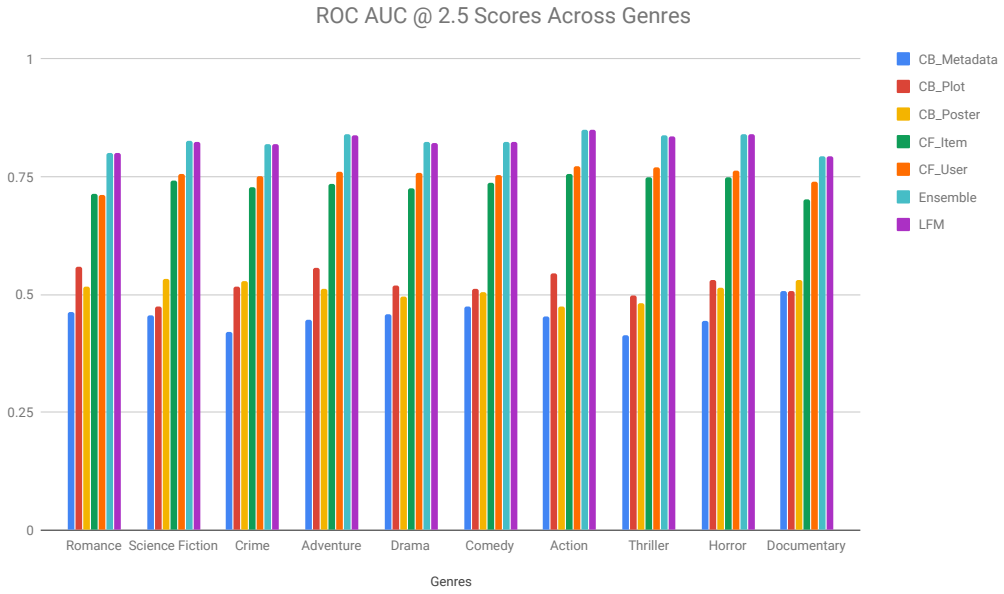

Figure 4: Comparison of ROC AUC @ 2.5 scores of different models across multiple genres. We can observe that the models perform consistently across various genres.

Table 3: Comparison of ensemble model performance.

Ensemble Models	ROC_AUC@2.5	RMSE@2.5
Ensemble-Overall	0.836120	0.677399
Ensemble-MovieSegments	0.836261	0.677246
Ensemble-UserSegments	0.836289	0.676205

lems, the future work would be to perform exhaustive feature selection to get a content-based model that works well, which would be studied under various scenarios and compared to the Collaborative Filtering models. In order to build content-based models on unstructured data like natural language and images (poster), a better feature embeddings could be obtained through semi-supervised methods based on deep learning techniques, which might help in improving their performance.

Moreover, the Collaborative Filtering models use Pearson Correlation in our project. It considers only the common set of movies watched by two users while taking similarity of two users, but does not consider the total number of movies watched by each user. This could be corrected by using a similarity score which is a product of Jaccard similarity (intersection over union) and the Pearson Correlation Coefficient. This could help in improving the Collaborative Filtering models.

7 Conclusion

So far, we have seen through our experiments, that the content based models have performed poorly. Even across segments of interest, we observe that the content based models give unsatisfactory results, for which the reasons have been mentioned in the limitations section. Feature representations of the plots and posters of the movies were not optimal, which caused the content based models to perform poorly. Therefore, we can conclude that the content-based models may require intensive feature selection to be effective.

However, the LFM and Collaborative Filtering (both CF-User and CF-Item) models showed better performance compared to the content-based models. Specifically, the LFM showed consistently better performance across all scenarios considered in this study. This could be attributed to supervised nature of the algorithm which learns optimized latent factors while considering the bias terms as well.

The ensemble model after training on the rating predictions obtained from each of our individual models, closely follows the predictions of the LFM model. Since ensemble model is hard to maintain as it requires multiple models to be trained and since the ensemble models are providing much lift in the performance, we can conclude that the LFM is the most suited for this data.

References

[1] Banik, Rounak. The Movies Dataset. 10 Nov. 2017, www.kaggle.com/rounakbanik/the-movies-dataset. Accessed 25 Apr. 2019.

[2] Tuinhof, Hessel, Clemens Pirker, and Markus Haltmeier. "Image-Based Fashion Product Recommendation with Deep Learning." International Conference on Machine Learning, Optimization, and Data Science. Springer, Cham, 2018.

[3] Bergamaschi, Sonia, and Laura Po. "Comparing lda and lsa topic models for content-based movie recommendation systems." International Conference on Web Information Systems and Technologies. Springer, Cham, 2014.

[4] Danilova, Viktoriia. “Hybrid Recommender Systems : The Review of State-of-the-Art Research and Applications.” (2017).

[5] (2008) Pearson’s Correlation Coefficient. In: Kirch W. (eds) Encyclopedia of Public Health. Springer, Dordrecht.

[6] (2011) TF–IDF. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA.

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3 (March 2003), 993-1022.

[8] Landauer, T., Foltz, P. & Laham, D. (1998). An introduction to latent semantic analysis. Discourse processes, 25, 259–284.

[9] Kiefer, J.; Wolfowitz, J. Stochastic Estimation of the Maximum of a Regression Function. Ann. Math. Statist. 23 (1952), no. 3, 462–466. doi:10.1214/aoms/1177729392. https://projecteuclid.org/euclid.aoms/1177729392.

[10] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[11] Jia, Yangqing, et al. "Caffe: Convolutional architecture for fast feature embedding." Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014.

[12] Arthur E. Hoerl and Robert W. Kennard. 2000. Ridge regression: biased estimation for nonorthogonal problems. Technometrics 42, 1 (February 2000), 80-86. DOI=http://dx.doi.org/10.2307/1271436.

[13] Altszyler, E., Sigman, M., Ribeiro, S., Slezak, D. F. (2016). Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. arXiv preprint arXiv:1610.01520.

[14] 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 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872