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 meta data, 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 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 approach to build such a hybrid model as an ensemble of various methods like item-based collaborative filtering, user-based collaborative filtering, Latent Factor Model, content-based model, Plot-text-based model and Poster-based model. We plan to explore how to combine the predictions from these models to get a final ratings for a user-movie pair. Also, we explore to understand how useful are each of the models to address each specific issue like cold-start problems for new user, unique tastes of user and recommending new/unpopular movies and recommendations for specific genres. This would result in a summary of the models which are effective in catering to specific issues. For this task, 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 pre-trained on a large product database and fine tune it for their dataset. The concatenated values from the second last layer of both of these networks is 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 due to which it wouldn't have popularity bias issue.We emphasize on the representation of the plot text detailed in this paper. Each plot text is preprocessed by removing stop-words and lemmatization the words, and vectorized using TF-IDF technique [6]. Further, dimensionality reduction using Latent Dirichlet Allocation (LDA) [7] and an Singular Value Decomposition (SVD) based approach called Latent Semantic Analysis (LSA) [8]. The similarity of the movies is assessed based on cosine similarity score taken over the topic space defined by LDA or LSA. The authors suggested that the cost of computation of LDA matrix is lesser compared to LSA (both time and space). However, during evaluation, the authors found the quality of topics and the recommendations generated using LSA to be better than using LDA. Hence, we choose LSA as the dimensionality reduction method for our plot text in our content-based model.

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 missing values in the matrix (test-set).
- **Model Evaluation**: 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 number of ratings that the movies have received and then study the performance of each model on each bin. In a similar fashion, we can perform binning on users based the number of movies that they have rated. Also, we evaluate the performance of each model on each genre. 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 6 and 5 respectively.

4 Data Description

For this project, we intend to use a single dataset consisting of 4 different components:

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. We store this matrix as a dictionary with the user being the key and the value being the list of all movie:rating pairs for the movies the user has rated.
2. **Movie Metadata Matrix**: This matrix consists of the details on every movie, such as, the release date of the movie, country and company of production, cast, crew, average rating and so on. There are 24 different features per movie and in all, 45466 different movies.

3. **Movie-Poster Content Matrix**: This matrix consists each movie along with its poster, with each poster represented as a flattened vector.
4. **Movie-Plot Content Matrix**: This matrix consists of each movie along with its plots, with the plots represented in terms of word embeddings.

4.1 Data Collection and Summary Statistics

The TMDB dataset contains a list of movies, with each movie having features like genre of the movie, its crew, cast, countries in which it was produced, date of release and so on.

Summary Statistics:

1. **genres**: There are 4096 different combinations of genres in the dataset for representation for each movie, with the genre 'Drama' being the most frequent one. There are in total 32 unique genres.
2. **original language**: There are 89 different languages in which movies are produced.
3. **original title**: Story,Night, 'la', Life, Last, 'Les', Dead, Christmas, House, World are some of the words which are the most frequent in the titles of the movies.
4. **popularity**: Here we see that minimum value for the popularity of a movie is 0, whereas the highest popularity for a movie is 547. It has a mean score of 2.92
5. **production companies**: There are 22708 different companies that have produced movies
6. **production countries**: There are 2393 different countries in which movies have been produced.
7. **release date**: There are 17336 different days on which movies have been produced.
8. **runtime**: On an average, a movie runs for 94 minutes. The longest movie in the dataset runs for 1256 minutes.
9. **spoken languages**: There are a total of 1931 different combinations of languages used as spoken languages in movies, with English being the most frequent one.
10. **vote average**: The minimum rating given to a movie is 0, while the maximum rating given to a movie is 10. On an average, a rating of 6 is given to a movie.
11. **vote count**: There are movies which have never been voted on. Hence the minimum vote count is 0. The maximum number of times a movie has been voted is 14075 times, where as on an average a movie has been voted on 10 times.

4.2 Data Preprocessing

We see that there is a lot of missing data in the dataset. Some features also do not contribute in any manner, which need to be removed. A lot of movies contain multiple spoken languages, they may classify under multiple genres and even might have been produced in different countries. For each movie, we plan to convert these features as one hot vectors. For example, if a movie comes under genres like Drama, Action and Comedy, we would represent the genre features as a one hot encoded vector with ones for Drama, Action and Comedy and zeroes for other genres. The popularity scores currently are not standardized. We would standardize these values for a better understanding of the feature. For the textual data, we would first need to remove the stop words and then generate the tf-idf or word embeddings vector.

4.3 Initial Findings from Exploratory Analysis

1. genres: We observe that majority of the movies were of the following genres: Drama, Comedy, Thriller, Romance and Action. The genre Drama accounts for almost half of the movies in the data set.
2. original language: Around 70% of the movies have been shot in the English language. English, French, Italian, Japanese and German languages account for around 85% of the movies in total.
3. popularity: Some of the top movies, ordered by their popularity scores are Minions, Wonder Woman, Beauty and the Beast, Baby Driver, Big Hero 6, Deadpool, Avatar and Guardians of the Galaxy.

4. production companies: The top production companies in terms of movies produced are: MGM, Warner Bros, Paramount Pictures, Twentieth Century Fox and Universal Pictures.
5. production countries: The top countries in which the maximum number of movies are produced are: United States of America, United Kingdom, France, Japan and Italy.
6. vote count: We see that famous movies and fan favourites like Inception, Dark Knight, Avatar, Avengers and Deadpool have received the maximum votes.

5 Challenges

- **Data Quality Challenges**: Whenever we encounter a null value in the data used by a content-based model, the model wouldn't be able to produce a recommendation for that entry. If this model is being used downstream by an ensemble model, we need to make sure that these missing values are not treated as a negative example.
- **Cold Start Problem**: One of the most common challenges of any recommendation system is the Cold Start problem. This can be with respect to both users and movies. When a new user joins the platform, the recommendation system has no prior data on the user's preferences and so recommending a movie to the user is more challenging. Similarly, when a new movie is released or added to the platform, we need to make sure that is it being recommended to the correct users since otherwise the movie will never be watched or rated. This is closely related to the popularity bias problem where we fail in recommending niche items to the users.
- **Curse of Combinatorics**: When evaluating the effectiveness of different models for recommending, there are cases where certain models work better than other for certain situations, for ex., content-based models might be better at predicting ratings for movies that have not been rated much. Since we have so many parameters and infinite ways to segment each parameter leading to infinite combinations to evaluate.
- **Large n, Larger d**: The size of our rating matrix is roughly 26 million entries. Holding it in a matrix form wouldn't be efficient. Also, in the case of Content-based recommendation, we plan to use features from the poster and the plot of the movie. It is easy for the size of this combined data to grow rapidly if not dealt with efficiently.

6 Proposed Methodology

We propose a hybrid model to predict the ratings for each user-movie pair. Firstly, we randomly hold-out 30% of the user-movie rating pairs as an unseen test set and train on the remaining 70% of the data. We build 6 different 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.

We use Pearson Correlation Coefficient [5] (since it doesn't penalize the missing ratings) as the similarity metric in all the relevant models (referred as sim_{ij}). The following are the 6 models which are being used in our method: The bias term (baseline model) included in the below equation b_{xi} is defined as the sum of the overall mean movie rating (μ), rating deviation of movie i (b_i), and rating deviation of user x (b_x).

- **Item-Based Collaborative Filtering (CF-I)**: It uses the Rating 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 ($N(i:x)$), where the weights are the similarity between the given item to each other item (S_{ij}), as shown in the equation 1.

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

- **User-Based Collaborative Filtering (CF-U)**: It uses the Rating 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 ($N(i:x)$), where the weights are the similarity between the given user to each other user (S_{xy}), as shown in the equation 2.

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

- **Latent Factor Model (LFM)**: It uses the Rating Matrix. We consider $k = 100$ factors, and factorize the Rating 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 (need to find optimal values).
- **Content-Based Model - Metadata (CB-M)**: 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***, weighed by the respective ratings. The rating prediction would be the similarity between the item-profile ***i*** and the user-profile ***x***.
- **Content-Based Model - Plot Text (CB-P)**: It uses the Movie-Plot Content Matrix, where plot text is represented as a *tf-idf* [9] vector for each movie. Also, dimensionality is reduced by applying SVD based LSA method with number of topics $k = 500$ discussed and suggested in section 2. If time permits, we would like to use word2vec [10] for the word embedding to get feature vector for each plot. Please note that the words from the title of the movies are separately embedded and concatenated to the plot vector. Now this problem is solved in a similar way as the CB-M model.
- **Content-Based Model - Poster (CB-P)**: It uses the Movie-Poster Content Matrix. Feature embeddings from Convolutional Networks [11] are used to represent the high dimensional poster images as vectors of lower dimension $k = 1024$ for each of the poster images. Now this problem is solved in a similar way as the CB-M model.

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

The ensemble of the above models would be created as a linear combination of the predictions from each of the above models and the corresponding weights would be learnt by training a Ridge Regression model [12] using SGD optimization algorithm [9] since it scales better for large datasets. Here, we would find the optimal ridge regularization coefficient by performing an hold-out cross validation on the train data.

Please note that the above-mentioned values of k (size of the reduced dimensional space) has been stated based on the results observed in reviewed literatures and prior experience. During implementation we would perform a hold-out cross-validation to optimal values. Similar approach is used to select best regularization coefficients for the above models.

In our implementation, we shall use a sparse representation for the Rating Matrix, and every matrix operation would ensure that the sparsity is maintained. We also plan to implement SGD in Apache Spark (by using the *mllib* library). Our analysis would include the comparison of time taken by each of the models to train, and for inference.

6.1 Evaluation Details

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

1. **Area under ROC curve** (Primary): We first binarize (whether a movie is recommended to a user or not) the ground truth ratings by setting **3** 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. **Root Mean Squared Error** (Secondary): The benefit of this metric is that we don't lose out on any information, but the drawback is that we penalize errors in all the levels of ratings equally.

7 Plan for Project Completion

The data matrices have already been created. We have also reviewed the methods and found necessary packages for feature embeddings. The collaborative filtering and content based models would be implemented by ourselves. We shall split the model implementations amongst ourselves and hence, each of us can implement 2 models each to complete 6 defined models. The ensemble training would be conducted after implementing these models. We plan to complete this much by May 20. Further, we shall perform exhaustive evaluation of the model based on the primary and secondary evaluation metric explained above, followed by comparing the performance in different scenarios as explained in the 3, which could be complete by the end of May. The report writing and poster creation would be completed by the respective submission deadline.

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.