Movie-based Recommendation System to Connect Users

Sumanth V Rao
Computer Science Dept.
PES University
Bangalore, India.
sumanthvrao@gmail.com

Sumedh Basarkod
Computer Science Dept.
PES University
Bangalore, India.
sumedhpb8@gmail.com

Suraj Aralihalli
Computer Science Dept.
PES University
Bangalore, India.
suraj.ara16@gmail.com

***Abstract*—General Movie Recommendation Systems identify the user preferences in movies, to recommend new movies for each user to watch. In this report, we present a novel idea of finding an ideal partner to watch a new movie with. Considering simple demographic details of individuals that comprise of age, gender, occupation, along with user's ratings and preferences in movies, the model looks for finding those ideal pairs with similar personalities who can go for a recommended movie together. We have achieved to recommend movies based on three different approaches along with users to watch them with.**

***Index Terms*—Recommendation System, Content-based Filtering, Collaborative Filtering, Surprise, Restricted Boltzmann Machine.**

I. Introduction

There has been substantial decline in the figures of people visiting theatres as a result of multiple reasons New York Film Academy Article. The prime reasons are lack of identifying the right movies that matches users interests. The already existing recommendation systems do a reasonable job in recommending similar movies based on one's interests, but fail in other aspects. These mainly include offering the right company for the user to watch a movie with. We address this problems, offering solutions to improve the overall situation. This indicates the need for further improvements to make recommendation systems more effective. Our model achieves this by pairing up the individuals to watch a recommended movie together. In order to do this, the model considers various other factors that include lot more than similar preferences in movies. Each movie has its own story-line that can interest only a subset of people and is not sagacious to gauge the movie on a common criteria. The model aims at alleviating this situation by identifying the right group of people to whom this movie should be recommended. Our focus hence shifts from the paradigm of suggesting movies to suggesting similar users.

II. Related Work

A. Prior Work

The entire model of identifying the ideal pairs is contingent on few basic assumptions that revolve around the conjecture that Movies play a significant role in defining the personality of the individuals who watch them. If we classify users based on general traits then teens or indignant individuals are expected to prefer action, adventure over fantasy or drama. Similarly young, bold individuals are likely to prefer romance, mystery genres over sorrow, tragedy. Users can be grouped into clusters based on various parameters like age, occupation that contribute significantly to user's personality and behaviour, which in turn play a significant role in their preferences for different genres.There are numerous research experiments, most of them documented vouch for this argument.

B. Background

Movie Recommended systems can be broadly classified into four important groups [3] Demographic Filtering, Collaborative filtering, Content-based filtering, and Hybrid filtering.

1) **Demographic Filtering** This is the naive way of filtering with limited accuracy. It mainly considers the available demographic details like gender,age,locality of residence,and profession of the users.
2) **Collaborative filtering** takes into account the interest of the user, by considering the interests of other similar users. The intuition behind this approach is that if user A and user B can be considered as having the same interests, it is reasonably to assume user A has also the same opinion about a new item only user B has already an opinion of.
   Instead of solely banking on item similarities, user similarities are also considered to make the recommendations more effective. Collaborative filtering can be further categorized into two sub categories: Memory-based collaborative filtering algorithms and model-based collaborative filtering algorithms.
   Collaborative Filtering considers all the available user-item information to make a prediction. Based on the available data,it determines the most related users, similar to target user.
   The strategy is to first develop a model of user ratings only.To achieve this it uses numerous machine learning techniques that include clustering Model-Based Collaborative Filtering clustering, rule-based and Bayesian network strategies.
3) **Content-based Filtering** The approach used in this filtering technique is to recommend movies to the user on the descriptions of previously evaluated movies. Hence, it recommends movies because they are similar to the movies that the user has showed interest in the recent past.
4) **Hybrid Filtering** This filtering technique relies on the fact that content-based and collaborative filtering characteristics are almost complementary. Combining the two strategies can result in enhanced performance and reliability.

III. Proposed work

A. Data

The dataset under consideration is offered by MovieLens [5]. It contains 100,000 ratings by 943 users on 1682 movies. Each user has rated at least 20 movies, ensuring consistency. The movies have been categorized into 18 genres with each movie falling into multiple categories. The movies also contain the IMDb link to the movie profile giving us further information on its background.

The user demographics play a key role in our case. Roughly 3/10 individuals are females and the average age of individuals is in 20's. Additional information about the users include their occupation and zip code which helps us factor in distance as a metric. The major data includes the ratings given by user for a movie. In addition, information about release date and time stamp give us ample information.

First sub-dataset has details of the movies with a unique movie-id, this id is same as the one mentioned in second sub-dataset. This dataset has ratings given by each user to all the movies he has watched. Each user has a unique user id, the third sub-dataset has simple demographic details like age, gender etc against all the user-ids.

Movie id	Movie title	Release Date	IMDb url	Genres

User id	Movie id	Rating	Time stamp

User id	Age	Gender	Occupation	Zip code

B. Approach

Most of the recommendation systems only look at recommending movies based on user preferences and talk nothing about connecting similar users together in any means. Our model aims at bringing individuals of similar entertainment preferences together to form an ideal pair who can go out to watch a movie together.User demographic details like age, gender, profession and location are known.

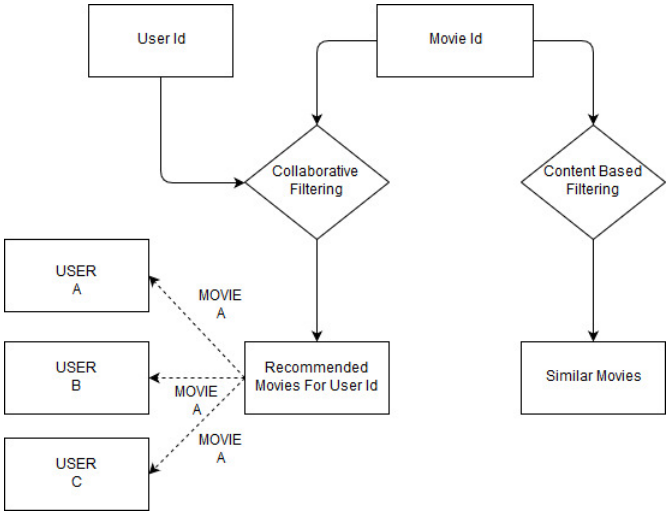

Fig. 1. Workflow Model summarizing our approach

In contrast to general recommendation systems, this model aims at considering numerous other insights of the movie.The dataset offers link to the movie description, So additional information related to the movie like the cast-crew, box-office collection, plot can be used to improve the overall accuracy of the model.

IV. Content Based Filtering

Content based filtering also referred to as cognitive filtering recommends items based on comparison between the content of items which means the items recommended by the model is same for any use. As we are building a recommendation system for movies, the content of each item is represented as set of descriptors or terms associated with the movie. Content-based filtering avoids the cold-start problem that forestalls other recommendation techniques, as the the system considers only the content of the movies to make recommendations, the recommended movies out-performs the quality of those early recommendations of other techniques that only become robust after millions of data points being added and correlated.

Content Based Recommendations rely on the characteristics of the item itself.The major challenge is in construing these characteristics of the item to be considered.The Original dataset consists of limited information about each movie.This data alone was insufficient to bring out valuable recommendations for a movie.To give you a brief idea about what characteristics a movie can pertain, let's look into the data the original data set comes with. Each movie in the dataset has details like movie title, year of release, movie id, imdb url and list of genres.The dataset classifies genres into 18 unique classes, where each class accounts for a different theme.Almost all the URLs were non-functional, leaving us with much lesser data than anticipated.

So we used tmdb (The Movie Database) api [**?**] to elicit more details for each movie.This api enabled us to obtain few other characteristics like names of the protagonists, director etc.We only considered the names of the primary two actors and the director in our feature set, because the rest of the cast and crew wouldn't be any significance.To add weight to the director of the movie we repeated the name 3 times. We created a hybrid feature for each movie which comprised of the name of the movie, year of release, list of genres, name of the director repeated 3 times, name the primary actor, name of secondary actor. After acquiring all the practicable features for each movie, We used a Countvectorizer to generate a matrix of movie versus word frequency. The Countvectorizer module identified 9105 distinct new features for each movie where each feature is a word extracted from the hybrid feature set of all the movies.We then calculated the cosine similarity of the matrix with itself to identify the similarity of one movie with every other movie in the dataset.Based on this similarity matrix we recommend 15 movies for every given movie.

V. Collaborative Filtering

Unlike content-based filtering which recommends movie based on "content" which in this case is Movie's features like genre, actors, directors etc, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The collaborative filtering model attempts to recommend movies and

how much a user likes each movie by considering the similarity of this user with other users.

A. Surprise Library

We used the Surprise (Simple Python Recommendation System Engine) library for Collaborative filtering. As the name suggests, it is a simple yet an excellent Python Scikit for recommender systems. It has a vast number of built-in algorithms which we made use of for predicting the ratings of those movies which the user hasn't watched.

We used the Surprise (Simple Python Recommendation System Engine) library for Collaborative filtering. As the name suggests, it is a simple yet an excellent Python Scikit for recommender systems. It has a vast number of built-in algorithms which we made use of for predicting the ratings of those movies which the user hasn't watched.

Some of the algorithms of the Surprise library that we tried and validated are Singular Value Decomposition(SVD), KNNBasic, and KNNWithMeans. KNNBasic, and KNNWithMeans. These require a similarity measure to estimate rating. We considered cosine similarity, and Pearson Baseline for estimating the similarity between users. To validate each of the above mentioned algorithms we made use of the 5-fold training and testing datasets; split being 80-20 across training and testing folds respectively. Each of these algorithms were used to predict the ratings for the testing folds, and were validated based on the mean RMSE(Root Mean Square Error), MAE(Mean absolute error) and training and testing time values across all the 5-folds. The KNN(K Nearest Neighbours) inspired algorithms were tested with k=40 neighbours. We chose SVD as our collaborative filtering algorithm as it had the least testing time, and low RMSE and MAE values across the 5-folds.

SVD sets the rating prediction for item i by user u as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

where, μ is the mean of all ratings, b_u is the user biases, b_i is the item biases, q_i is the item factors, and p_u is the user factors

If user u is unknown, then the bias b and the factors p are assumed to be zero. The same applies for item i with bi and qi. To estimate all the unknown, SVD minimizes the following regularized squared error:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \right)$$

The minimization is performed by a very straightforward stochastic gradient descent:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned}$$

where the error term is the difference between the actual rating and predicted rating. These steps are performed over all the ratings of the trainset and repeated 20 times. Baselines are initialized to 0. User and item factors are randomly initialized according to a normal distribution. The learning rate γ is set to 0.005 and the regularization term λ is set to 0.02

TABLE I
COMPARING SURPRISE'S BUILTIN ALGORITHMS FOR OUR DATASET

Algorithm	Mean RMSE	Mean MAE	Mean Fit time	Mean Test time
SVD	0.9358	0.7375	11.38	0.45
KNNBasic (pearson_baseline)	1.0005	0.7917	5.65	10.91
KNNBasic (MSD)	0.979	0.7731	1.23	8.47
KNNBasic (cosine)	1.0174	0.8045	4.41	9.26
KNNWithMeans (pearson_baseline)	0.9382	0.731	4.5	8.74
KNNWithMeans (MSD)	0.9502	0.7486	1.34	9.71
KNNWithMeans (cosine)	0.9556	0.7546	4	8.55

B. Restricted Boltzmann Machines

Salakhutdinov, Mnih and Hinton in their paper [6] talk of a novel idea in using a two layer undirected

Restricted Boltzmann Machine(RBM) for Collaborative Filtering. An RBM is a stochasic Neural Network. They are restricted in the sense that no two nodes in the same layer are connected together. The first layer of visible units are user's movie preference while the second layer of hidden units are the latent factor we are trying tp learn. A bias unit to account for the different popularity's of each movie is added. It is shown that an RBM slightly outperform carefully tuned SVD models [6].

The fundamental idea here is to use an RBM for each user with shared weights for users rating the same set of movies. Every RBM has the same number of hidden units, but an RBM has active softmax visible units only for the items rated by that user. If two users have rated the same movie, their two RBM's must use the same weights between the softmax unit for that movie and the hidden units. To ensure binary mappings, nodes with ratings from 1 to k are made for every user's RBM for each movie he/she has rating. Each node is activated or deactivated based on the value it is looking for. The first step is to make visible units K-nary to convert to a binary input.

Node 1 is looking for rating 1 for movie 1
Node 2 is looking for rating 2 for movie 1
. . . . . .
Node K is looking for rating k for movie 1
Node K+1 is looking for rating 1 for movie 2
. . . . . .
(and so on).

Missing ratings in the input are not dealt with. The whole idea behind an RBM is that it can reconstruct the input. In the training process we ignore these reconstructions.

The training process includes feeding in each training example by setting the visible units to the movies watched by him/her. We update the state of the hidden layer units by applying the logistic function on its activation energy.

$$a_i = \sum_i w_{ij} x_i$$

Based on this we compute a positive and negative factor for each edge. Units that are positively connected to each other try to get each other to share the same state, while units that are negatively connected to each other prefer to be in different states.

$$Positive(e_{ij}) = x_i * x_j$$

$$Negative(e_{ij}) = x_i * x_j$$

Based on these factors we update the weight of the matrix. Here L is the learning rate set by the user.

$$w_{ij} = w_{ij} + L * (Positive(e_{ij}) - Negative(e_{ij}))$$

This weight updation rule is called contrastive divergence. This objective function replays on Markov Chains and is similar to an approximate gradient descent algorithm. During this calculation we have to use Gibb's sampling to approximate the distribution and stop after k samples itself. This is repeated over every training example in the dataset. The final weight matrix built is saved and is used to reconstruct the input, which will predict the ratings for movies that the input hasn't watched, which is what we use to recommend movies.

VI. Results

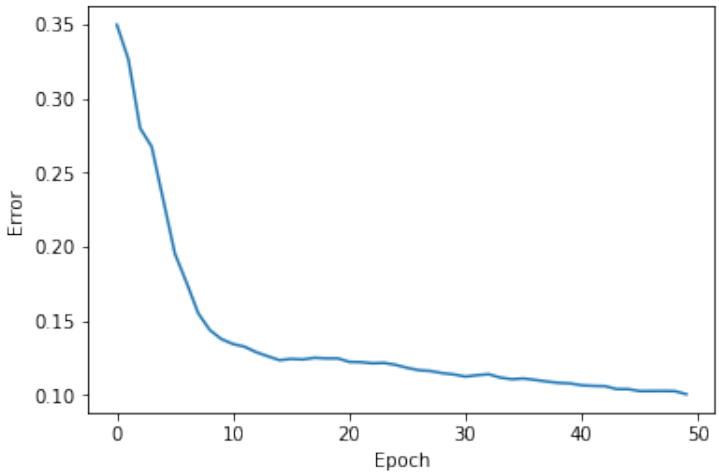

Fig. 2. Calculating Optimum epoch value during training for RBM

So it can be inferred from the graph that the optimum epoch value to be chosen for training RBM model is 50

So our end-product is a platform which expects user id, and an optional movie id. If the user enters just the user id, he/she is shown the top movies which he may like(is probable to give a high rating).

On the other hand, if the user enters both the user id as well as the movie id, he is shown top movies as before, and also similar movies to the movie id he entered. In addition to this, he is shown with all the users who are going to rate this movie same as the user. The user can now filter among these users based on their age, gender, and occupation and can now choose whom he/she wants to go watch this movie with.

VII. Conclusion and Future Work

Our recommender system is able to predict the ratings of users with an RMSE of 0.93 and MAE of 0.74. This is the crux of our recommender model, as this predicted ratings are mainly used for suggesting users with similar interest for a particular movie.

Our recommender system coupled with the novel idea of suggesting similar users to go out for a movie could be incorporated in movie booking websites such as BookMyShow which we believe could be a great hit among youngsters especially.

This recommender works well for already existing users who have rated movies. Future work is to recommend movies(and users)for new users as well, based on his/her demographic details.

References

[1] S.M.Bowes, A.L.Watts, T.H.Costello, B.A.Murphy , Scott.O.Lilienfeld. " Clarifying the role of abnormal and normal personality in music and movie interest ", 27 February 2018.

[2] L. Haesung, K. Joonhee. "Similar User Clustering based on MovieLens Data Set ", Advanced Science and Technology Letters, Vol.51, pp.32-35, CES-CUBE 2014.

[3] Steven Postmus. "Recommender system techniques applied to Netflix movie data", Vrije Universiteit Amsterdam, February 2018.

[4] C.Christakou, A.Stafylopatis. "A hybrid movie recommender system based on neural networks", ISDA'05, 8-10 Sept, 2005.

[5] MovieLens Dataset. "https://grouplens.org/datasets/movielens/100k/".

[6] R.Salakhutdinov, A.Mnih, G.Hinton. "Restricted Boltzmann Machines for Collaborative Filtering", Proceedings of the 24th International Conference on Machine Learning , Corvallis, OR, 2007.

VIII. Individual Contribution

Suraj Aralihalli : Data Preprocessing, Fetching Additional Data from APIs, Developed Content Based Model for Movie Recommendation

Sumedh Basarkod : Data Preprocessing, Statistical and Time Consumption Analysis of various Collaborative Filtering Algorithms, Collaborative filtering model using SVD algorithm, Worked on End Interface.

Sumanth V Rao : Data Preprocessing , Worked on Restricted Boltzmann Machine Approach of Recommending Movies, Developed End Interface,