ISyE 6740 – Spring 2023
Project Proposal

Team Member Names: Shaya Shakib

Project Title: Exploring the Use of Recommender Systems

Problem Statement

Recommender systems frequently give users personalized suggestions based on their interests and activities and are widely used in different industries such as finance, health, and entertainment. There are many challenges with data sparsity, cold start, scalability, and diversity that affect the results of the recommender system and are common among many available approaches.

Using the MovieLens dataset, which contains movie ratings and metadata of movies, I aim to explore different methods for building a recommender system and create a hybrid recommender system using the combination of those methods.

In this project, my focus is on comparing different techniques individually and then combining them to have a hybrid(ensemble) approach to improve the overall results of a recommender system. The methods that I will explore are collaborative filtering, content-based, and matrix factorization to increase the accuracy of movie recommendations.

My recommender system's effectiveness will be assessed using measures such as root mean square error (RMSE) and hit rate. I anticipate that a hybrid strategy that makes use of multiple approaches can produce superior outcomes to a single technique. Although the hybrid approach is not unique, I think this will enhance the result and offer valuable information to movie fans.

Data Source

GroupLens Research has made several datasets available under the name MovieLens. These datasets include reviews and tag applications for movies that have been gathered from the MovieLens website (http://movielens.org) over a variety of time periods. The datasets are available in various sizes, enabling researchers to select the dataset that best suits their needs by offering various information on user behavior and preferences.

MovieLens dataset that I am going to use in this project contains 100,004 ratings and 1,296 tag applications across 9125 movies. These data were created by 671 users between January 09, 1995 and October 16, 2016. This dataset was generated on October 17, 2016.

MovieLens Small dataset contains several files, as follows (Harper, 2015):

links.csv: This file contains identifiers that can be used to link the MovieLens data with data from other sources. Each row represents one movie and contains the following columns: movieId, imdbId, and tmdbId.

movies.csv: This file contains information about the movies in the dataset. Each row represents one movie and contains the following columns: movieId, title, and genres.

ratings.csv: This file contains the ratings given by users to movies. Each row represents one rating and contains the following columns: userId, movieId, rating, and timestamp.

tags.csv: This file contains the tags applied by users to movies. Each row represents one tag application and contains the following columns: userId, movieId, tag, and timestamp.

Methodology

As was mentioned in the problem statement, the aim of this project is to explore and compare different methods and finally create a hybrid recommender system. Here is a description of each method for this project proposal and more details will be provided in the final report.

Content-based filtering: In this approach, the recommender system recommends items based on the features of the items. In this approach, the recommender system, recommends movies to users based on genres.

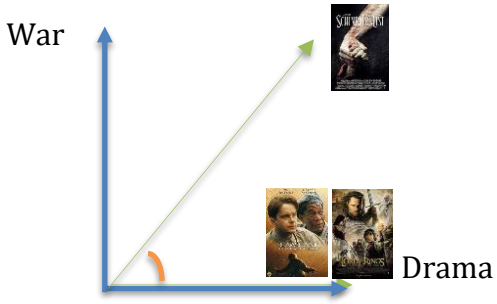
Movie	Genres
Schindler's List (1993)	Drama|War
Shawshank Redemption, The (1994)	Crime|Drama
Lord of the Rings: The Return of the King, The (2003)	Action|Adventure|Drama|Fantasy

The similarity between the movies can be calculated by converting genres to dimensions and then computing the cosine similarity:

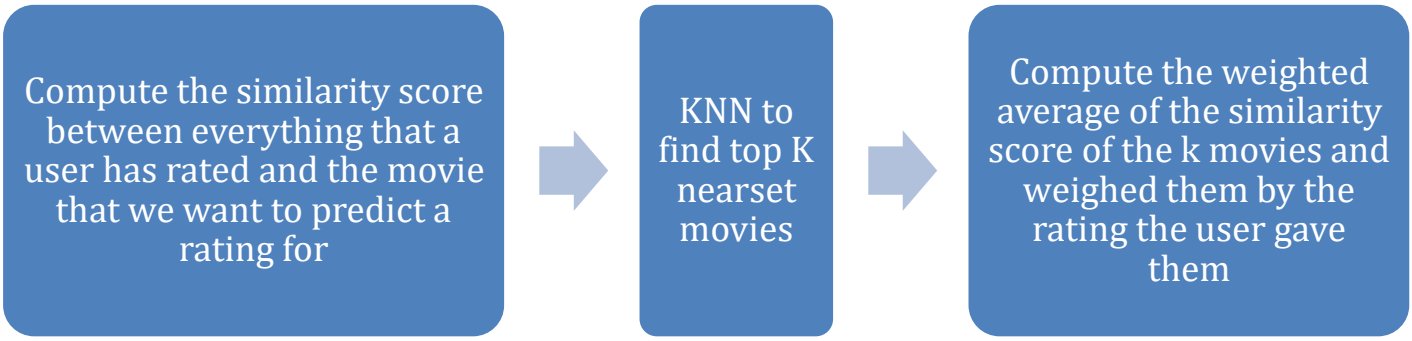
Movies	Drama	War	Crime	Action	Adventure	Fantasy
Schindler's List	1	1	0	0	0	0
Shawshank Redemption	1	0	1	0	0	0
Lord of the Rings	1	0	0	1	1	1

To calculate the multi-dimensional cosines, the following formula (Cosine similarity, n.d.) will be used:

$$CosSim(a,b) = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}}$$



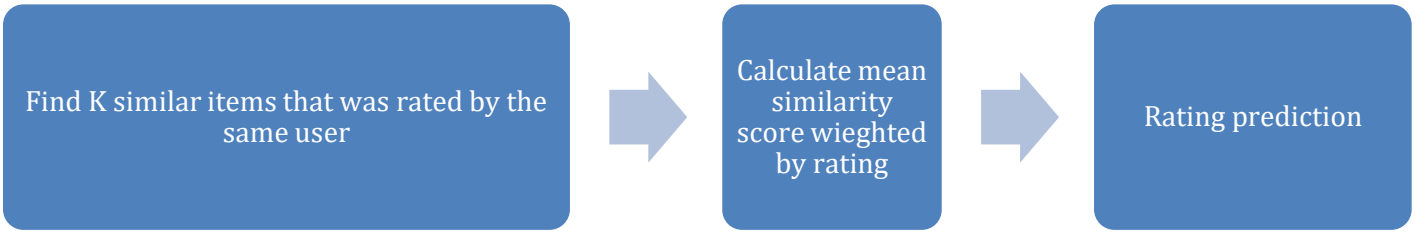
To calculate the rating, first, the similarity score between every movie that a user has rated and the movie that the rating is desired will be computed, and then, using the KNN approach, K movies with the closest similarity score to the movie that is being evaluated for the user will be selected. Finally, the weighted average of the similarity score of the K movies will be computed and weighed by the rating the user gave them. (Kane, 2020)



Collaborative filtering: As we know this approach can be done for User-User or Item-Item. User-user collaborative filtering calculates the similarity between users and recommends items based on the behavior of a user's nearest neighbors. In other words, it finds users who are similar to a given user and recommends items that similar users have liked. Item-item collaborative filtering, on the other hand, calculates the similarity between items and recommends items that are similar to those previously liked by the user. (Bart Baesens, 2020)
In this project, the focus is on the movie-movie (item-item) approach as items are simpler and belong to a smaller set(genres). Furthermore, movie similarity is more meaningful as they tend to not change over time. (University, Collaborative Filtering, 2016)
To compute the rating, for unknown values, first, we define similarity of movie i and j as S_{ij}. In this project, centered cosine similarity which is essentially a variation of cosine similarity where the attributes vectors are normalized by subtracting the vector mean will be used. Then k-nearest neighbors will be selected and rating $\widehat{r_{ui}}$ (how much user u likes movie i) can be estimated as follow:

$$\widehat{r_{ui}} = \frac{\sum_{j \in N(i,u)} S_{ij} \cdot r_{uj}}{\sum_{j \in N(i,u)} S_{ij}}$$

Where N(i, u) is set of movies similar to movie i that where rated by user u. (Kane, 2020)



Latent factor models: Suppose R is the rating matrix with rows are movies, columns are users, and values are rating scores, the main idea of this method is to use matrix factorization and SVD ($A = U\Sigma V^T$), to reduce the dimensionality of the user-movie matrix(R) and find the latent factors that can explain users ratings for different movies with the focus of reducing the discrepancy between the predicted value and true value.
This method essentially will resolve the recommendation problem via optimization by using SVD and stochastic gradient descent (SGD). Assuming that matrix R can be written as $R \approx Q \,.\, P^T$ (factorized R into two matrices) we can think of this as:

$$R_{movies \times users} = Q_{movies \times factors} \,.\, P^T_{users \times factors}$$

To estimate the missing rating, we can compare the above with SVD ($A = U\Sigma V^T$), and say:
$A = R$, $Q = U$, $P^T = \Sigma V^T$

SVD returns minimum reconstruction error (SSE) and SSE and RMSE are monotonically related, therefore SVD is minimizing RMSE, but since the R matrix has missing values, SVD is not defined when entries are missing. To find P and Q the following optimization needs to be solved:

$$\min_{P,Q} \sum_{(i,u)\epsilon R} (r_{xi} - q_i . P_u^T)^2$$

In summary in the above optimization problem, the goal is to find P and Q such that by summing over all the known ratings $(i,x)\epsilon R$ the square value of the rating r_{xi} minus our predicted rating $q_i . P_u^T$ are as small as possible. To solve the above optimization problem, gradient descent can be used(University, Latent Factor Recommender System, 2016). More information on how to compute this optimization problem using SGD will be provided in the final report.

Hybrid System (ensemble approach): A hybrid recommender system can be built using different methods and, in this project, a weighted hybrid method will be used. In the weighted hybrid approach, different weights are being assigned to the predictions of each individual method and they are being combined linearly to produce a final prediction. The weights can be based on different criteria, and in this project, accuracy, hit rate and manually tuned weights (trying different weights and report the best result) will be discussed. (Mr.Avadhut D.Wagavkar, 2017)

$$r_{u,i} = \frac{\sum \beta . r_{u,i}}{\sum \beta}$$

Evaluation and Final Results
There are various methods to evaluate a recommender system such as MAE, RMSE, Hit-Rate, ARHR, cHR, …. In this project, the focus is on RMSE and hit rate.

As we know the lower RMSE means a more accurate prediction and this method is commonly used for offline evaluation of recommender systems, where historical data is used to test the

performance of a model before deploying it to real users. However, RMSE does not capture other aspects of recommendation quality, such as diversity, novelty, or serendipity. The RMSE for our user/movie rating matrix R can be calculated as follows:

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{(i,u) \in R} (\widehat{r_{ui}} - r_{ui})^2}$$

Where $\widehat{r_{ui}}$ is the predicted rating and r_{ui} is the true rating of user u on movie i.

To avoid a narrow focus on accuracy and the fact that RMSE might penalize a method that does well for high ratings, in this project Hit rate and leave one out cross validation will be considered. To calculate hit rate, we generate top end recommendations for all the users in test set, if one of the recommendations in a user's top end recommendation is something they actually rated, it will be considered as a hit. (Users found something that is interesting enough to watch on their own). By adding all the hits in our top-end recommender system and dividing it by number of users we have hit rate.

$$hit - rate = \frac{hits}{Users}$$

It is expected that after tuning the weights of the hybrid approach that combines several methods, the results for RMSE or hit rate will be better than using just one method.