Movie Recommendation

Semester project report for Data Mining, Dr. Xiaofei Zhang instructing

Ramita Maharjan, U000769429
Department of Computer Science, University of Memphis, Memphis, TN, rmhrjan1@memphis.edu

ABSTRACT
With the plethora of movies and content in the internet resources, the viewers often feel perplexed about which movie to watch. Had the recommendation system not been implemented, the users would waste so much of their time trying to find the right movie for them to watch. The recommender system can predict whether a particular user would prefer a movie or not and hence is beneficial to both movie service providers and users. Not only does it save the viewer's time, but it also satisfies them by suggesting movies they like and hence increases the number of movies watched by the users. The movie suggestions can be based on the user profile and preferences or the similarity between users and movies. In this project, unsupervised learning algorithms (Clustering and Collaborative filtering) are used for recommending movies, and a supervised learning algorithm (Regression) is implemented to predict a user rating for an unseen movie.

1 Introduction

Wikipedia defines a recommendation system as a subclass of information filtering systems that seek to predict the 'rating' or 'preference' that the user would give to an item. A movie recommender system is an algorithm that aims at suggesting relevant movies to users. Today, every company uses some type of recommendation system and companies like Amazon, YouTube, Netflix, and others are doing a great job suggesting movies to users. According to Business Insider, the personalized recommendation engine saves Netflix $1 billion every year.

Recommendation systems can be of the following types:

i. Collaborative filtering: It recommends movies by identifying other similar users or movies. The user-based collaborative filtering technique calculates the similarity between users by comparing their ratings on the same movies whereas the item-based technique uses the similarity between movies.
ii. Content-based filtering: It is a domain-dependent algorithm, and it emphasizes more on the analysis of the attributes of movies to generate predictions.
iii. Hybrid filtering technique: It combines both collaborative and content-based filtering. The idea behind hybrid techniques is that a combination of algorithms will provide more accurate and effective recommendations than a single algorithm as the disadvantages of one algorithm can be overcome by another algorithm [1].

The DataSet: MovieLens

GroupLens is a research lab in the Department of Computer Science and Engineering at the University of Minnesota specializing in recommender systems, online communities, mobile and ubiquitous technologies, digital libraries, and local geographic information systems. Their objective is to advance the theory and practice of social computing by building and understanding systems used by real people [2]. The MovieLens 25M dataset is a publicly available stable benchmark dataset generated on November 21, 2019. It contains 25000095 ratings and 1093360 tag applications applied to 62423 movies by 162541 users. It also includes tag genome data with 15 million relevance scores across 1,129 tags [4]. The dataset describes a 5-star rating scale, with half-star increments.

The dataset contains .csv files for genome scores, genome tags, links, movies, ratings, and tags [3]. For this project, we have only used ratings.csv and movies.csv.

Ratings Data File Structure (ratings.csv)

All ratings are contained in the file ratings.csv. Each line of this file after the header row represents one rating of one movie by one user and has the format: userId, movieId, rating, timestamp. Ratings are made on a 5-star scale, with half-star increments (0.5 stars - 5.0 stars). Timestamps represent seconds since midnight Coordinated Universal Time (UTC) of January 1, 1970 [5].

Movies Data File Structure (movies.csv)
Movie information is contained in the file movies.csv. Each line of this file after the header row represents one movie and has the following format: movieId, title, genres [5]. Genres are a pipe-separated list, and are selected from the following:
Action
Adventure
Animation
Children's
Comedy
Crime
Documentary
Drama
Fantasy
Film-Noir
Horror
Musical
Mystery
Romance
Sci-Fi
Thriller
War
Western
(no genres listed)

2 Related Works

Several research works have been done in building a movie recommendation system using the MovieLens dataset. In July 2019, Khishigsuren Davagdorj and co-authors have published their conference paper entitled "A Collaborative Filtering Recommendation System for Rating Prediction" using Biased Matrix Factorization and a regular Matrix Factorization, both using Stochastic Gradient Descent (SGD) [6]. In SIGKDD-2020, Yifei Ma and other authors' paper entitled "Temporal-contextual Recommendation in Real Time" was announced as the best paper in the applied data science track. The HRNN-meta recommender model achieves state-of-the-art performance on various datasets [7].

Chavare et. Al in their paper "Smart Recommender System using Deep Learning" have used the MovieLens dataset for recommendation [8]. Saman Forouzandeh and other authors in their paper "Presentation of a recommender system with ensemble learning and graph embedding: a case on MovieLens" have combined ensemble learning, the fuzzy rules, and decision trees and then used a knowledge graph for recommendation [9].

In 2021, Si Thin Nguyen and co-authors have published their research paper entitled "Featured Hybrid Recommendation System Using Stochastic Gradient Descent" which focused on a novel incremental algorithm using in featured hybrid system combing the feature of the content-based method and the robustness of matrix factorization in collaboration filtering [10]. The hybrid approach of the recommendation engine is better because it combines the pros of both content-based and collaborative filtering mechanisms.

3 Problem Statements

The goal of this project is to build a simple recommender system for movies using MovieLens 25M dataset that can suggest movies liked by most similar users and predict user rating of movie that has not been watched by the user.

3.1 Supervised Learning

The supervised learning problem addressed in this project is that of Movie Rating Prediction, defined as the following:

Movie Rating Prediction

Instance: Genres and year of release for movies
Output: Given a user ‘U’ and a movie id ‘M’ unrated by ‘U’, what rating will ‘U’ give to ‘M’?

In other words, the instance, in this case, is a whole collection of movie ratings by users and movie details. The goal is to predict a rating value between 0 and 5.

3.2 Unsupervised Learning

The corresponding unsupervised learning problem addressed here is that of Movie Recommendation, defined as follows:

Movie Recommendation
Instance: Movie ratings by users and movie genres
Output: Given a user id, which users are similar to that user?

The instance, in this case, is a whole collection of movie ratings including movie genres. The idea is to divide the users into different clusters and find the most similar users to that user.

4 Solving the Problems

In other to solve the problems listed above, we have used Data Mining algorithms. We will provide some details concerning the Data Mining models designed to address each problem and then assess how well these models performed using some evaluation metrics. For development purpose, we used Jupyter Notebook platform and Python as the programming language.

Data Pre-processing:

The first step performed for the data mining algorithms is data preprocessing. We first gathered 25000095 total ratings from the ratings file and considered only the userId, movieId, and rating attributes and omitted timestamp. Since we are going to recommend the movies that are liked by most users, we filtered only those rating records that have at least a 3.5 rating. We also filtered records containing ratings for movies that are mostly rated by users. Because this dataset is huge and considering the limitation of the computing machine, we have sampled from ratings to only comprise a 0.2% of those ratings. The next step done is simple data cleaning where we checked for NaN values and dropped such records if they have no “rating” or “genres” values. The result of this step is 349939 ratings for 731 movies by 96780 users.

4.1 Movie Recommendation

All the 349,939 ratings from the preprocessing steps are then divided into training and testing datasets in a 7:3 ratio. To suggest movies to a user, the idea is to group similar users into clusters. There were around 20 genres, and a movie was associated with one or more genres (separated by |). Firstly, we grouped all the ratings for a user and calculated the average genre ratings per movie genre so that each user has a single record for the average genre ratings. Now, the training dataset contains 70% of 96780 average ratings per user and the rest are used for testing.

We have implemented clustering algorithms using a Python package called scikit-learn. KMeans clustering is the most popular clustering algorithm which requires the number of clusters to be specified by the user. For this purpose, the Elbow method is used, and the value of ‘k’ given by the method is passed as the number of clusters to KMeans.

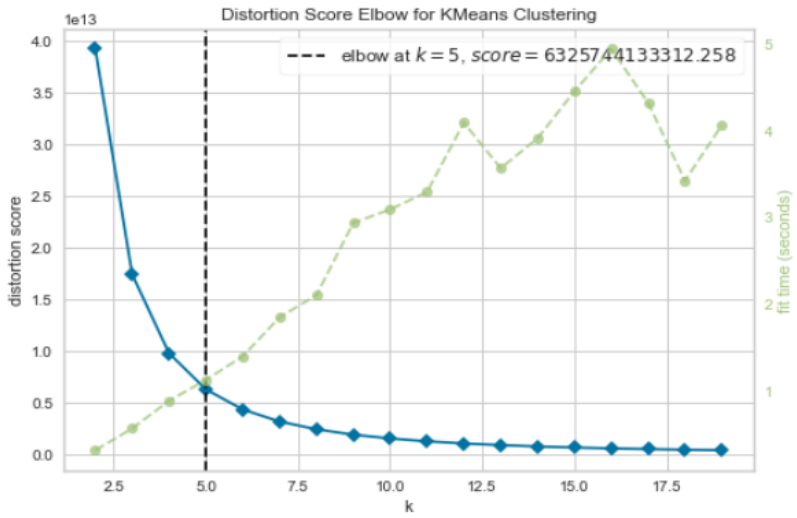

Figure 1: Elbow method for finding k for KMeans Clustering

We have also used BIRCH clustering to which the number of clusters is passed the same value of k given by the Elbow method and MeanShift clustering which only takes the value of the bandwidth of the window. The result of the clustering step is summarized in the table below:

Cluster	Kmeans		BIRCH		MeanShift	
	No of users	No of movies	No of users	No of movies	No of users	No of movies
0	19530	651	25656	675	25058	686
1	19216	658	22453	660	36643	683
2	19536	670	18555	659	35079	684
3	19317	663	19604	662		
4	19181	661	10522	620		

Table 1: Count of users and movies in clusters using different Clustering algorithms.

After finding the cluster for the users, the next step in the recommendation process is finding the most similar users within that cluster. The 25 nearest users in similarity are listed using the Cosine similarity between users based on the average rating they have given to the movies measured in terms of their genres. Now, with 25 most similar users, all the movies that are highly rated by those users are listed from which the movies that have already been rated by the given user are removed. The remaining movies are provided as a list of movie titles along with the year of release in parenthesis and the genres of those movies and then recommended to the user. Here, the underlying idea is that a user would most probably like a movie if it is highly rated by users who have liked movies with similar genres.

4.2 Movie Rating Prediction

We have applied regression technique to predict the rating of a user for a movie that the user has not watched. We have used the ratings from pre-processing step and extracted the year of movie release from the movie title. Since the movie genres were a single string containing different genres separated by '|', we converted it to numeric format by using One-Hot Encoding technique. This technique gives a '1' if the movie has that genre and a '0' if the genre is not present. The dataset is then divided into training and testing set such that the testing set contains 33% of the data. Again, the Python package scikit-learn is used for the regression model. 236682 rating data is used for training the linear regression model and the remaining 116575 ratings are used for testing the model. Similarly, we have built a regression model using ridge and lasso regression technique as well and calculated root mean square error and R squared for evaluating the models. The following table summarizes the evaluation metrics of these three models:

Regression Model	Root Mean Square Error (RMSE)	R Squared/ Coefficient of Determination
Linear regression	0.523	0.024
Lasso regression	0.526	0.012
Ridge regression	0.523	0.024

Table 2: RMSE and R-Squared values for different Regression models

Once the prediction model is built, we chose a userId 'U' and a movieId 'M' randomly from the dataset such that 'M' has not been rated by the user 'U'. This movieId is then used to extract the movie details from which the year of the movie release and the one-hot encoding for the genres are passed to the prediction models. Given below are some of the results for movie rating prediction.

UserId	MovieId	Linear Regression	Ridge Regression	Lasso Regression
91914	2347	4.29	4.29	4.27
160279	1707	4.15	4.15	4.23
84745	2401	4.26	4.26	4.26
114745	1952	4.34	4.34	4.31
69364	1730	4.25	4.25	4.23

Table 3: Movie rating prediction by different users using different Regression models

Since the RMSE and R-squared values are better for linear and ridge regression models, we can use any of these two models for predicting the rating of a movie not rated by the user.

5 Discussion

Out of six .csv files in the dataset, we used only two and both the solutions are based mainly on the ratings, genres, and release year of the movies. The results could vary and probably be better representation if we have taken the tags and IMDB ratings into account as well. Since sampling of the dataset takes random rows from the dataset, the dataset changes every time. It is a time-consuming process to load the datasets and perform operations on them again and again. So, we saved the results of the sampling and preprocessing into the disk. These saved files are then used as input in clustering steps next. It took 5 to 8 seconds to load the 25M dataset ratings.csv in our i7 11th gen 12 GB machine.

The training and testing datasets are also saved in local files for easy access and saving time. BIRCH and MeanShift clustering took significantly more time than KMeans for performing clustering and KMeans also grouped the users in similar size clusters. So, the results of KMeans clustering are also saved into one .csv file per cluster. Similarly, the regression models are also saved. A very important thing to note is that the results and outcomes of the models used vary with the change in the sampled dataset, size of the dataset, and choice of training and testing datasets.

Due to hardware and time constraints, some clustering algorithms like agglomerative clustering could not perform well on the dataset we have pre-processed. Based on my observation, it took a very long time for encoding the movie genres using One-Hot Encoding and merging them with the movie ratings data from the preprocessing step, and save them into a file.

It would certainly be better if we can try more models for clustering, similarity calculation, and regression and choose the best ones. For future works, we would like to consider all the data files provided in the dataset. The results would be a better representation of the whole dataset if we can perform these tasks on a machine with enough specifications

for 25M dataset. Overall, this project was a great learning experience that brought us a step closer to working in the real dataset for a real-world data mining problem.

References:

1. F.O. Isinkaye, Y.O. Folajimi, B.A. Ojokoh, Recommendation systems: Principles, methods and evaluation, Egyptian Informatics Journal, Volume 16, Issue 3, 2015,Pages 261-273
2. https://grouplens.org/about/what-is-grouplens/
3. https://grouplens.org/datasets/movielens/
4. F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4: 19:1–19:19. https://doi.org/10.1145/2827872
5. https://files.grouplens.org/datasets/movielens/ml-25m-README.html
6. Davagdorj K., Park K.H., Ryu K.H. (2020) A Collaborative Filtering Recommendation System for Rating Prediction. In: Pan JS., Li J., Tsai PW., Jain L. (eds) Advances in Intelligent Information Hiding and Multimedia Signal Processing. Smart Innovation, Systems and Technologies, vol 156. Springer, Singapore. https://doi.org/10.1007/978-981-13-9714-1_29
7. https://www.kdd.org/kdd2020/accepted-papers/view/temporal-contextual-recommendation-in-real-time
8. S. R. Chavare, C. J. Awati and S. K. Shirgave, "Smart Recommender System using Deep Learning," *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, India, 2021, pp. 590-594, doi: 10.1109/ICICT50816.2021.9358580.
9. Forouzandeh, S., Berahmand, K. & Rostami, M. Presentation of a recommender system with ensemble learning and graph embedding: a case on MovieLens. *Multimed Tools Appl* **80,** 7805–7832 (2021). https://doi.org/10.1007/s11042-020-09949-5
10. https://doi.org/10.2991/ijndc.k.201218.004