Northeastern University

Final Project Research Paper

Course: Data Science Engineering Methods
Academic Year: Spring 2019

Professor: Mr. Nicholas Brown

Submitted by:
Shardul Pathak
Mihir Sawant
Sarthak Goel
Submission Date: 04/26/2019

Movie Recommendation System

Mihir Sawant
Sarthak Goel
Shardul Pathak

Northeastern University
INFO 6105 Data Science Engineering Methods and Tools
Spring 2019
Professor Nicholas Brown

ABSTRACT

This research is about the implementation of different algorithms on Movielens dataset and recommending movies to users. We have employed different filtering techniques initially to check how well they perform on the dataset. Since Collaborative filtering is the most widely used technique for recommendation systems, so we have also implemented the same. Each technique has its own drawbacks which is the purpose of this research project. We have implemented an algorithm that uses a pre-filter before applying K-means clustering algorithm. Inputs which are related to different attributes of a movie are provided by a user to filter out the results that contains the movies. Each of the attribute of a movie has a different weightage in recommendation of a movie. The filtering of movies is followed by application of the pre-filter that matches the attribute values to the weightage that is provided for an attribute and then the total weightage will be computed before application of k-mean clustering.

INTRODUCTION

Data has become one of the most essential parts in the operations of a business. It helps us understand what the user wants because of which a business can provide those services so that both the user and business can benefit from them. Users of a system are unknowingly playing a very important role in providing this data for the database on which analytical models are built to understand the user's requirements. Drawing the information from a user is used for helping users and providing them with better services.

Recommendation systems are the most talked topic in the business now a days as it helps in catering the customer requirement in a better way by providing better customer experience and at the same time it helps business to grow by satisfying the customer requirements in the best possible way. These recommendation systems are used in various industries like education, ecommerce, job boards, travel, real estate etc. A well-designed recommendation system will compute the user taste for a product and recommend the user similar with similar products.

Movielens is an online platform that uses a recommender system to recommend movies to users using collaborative filtering. Being a popular recommender system, we are using the dataset of Movielens which was collected by the GroupLens research project at the University of Minnesota.

Different filtering techniques are used for recommending movies to users. Among all the available techniques, collaborative filtering is the popular and efficient filtering technique that is also employed by Movielens for recommending movies to users. The CF based 'movielens' recommender system uses several algorithms such as user-based, item-based and SVD. These filtering techniques have their major drawback which is the cold-start problem as the algorithms fail to provide results without historical data.

Content Based Filtering:

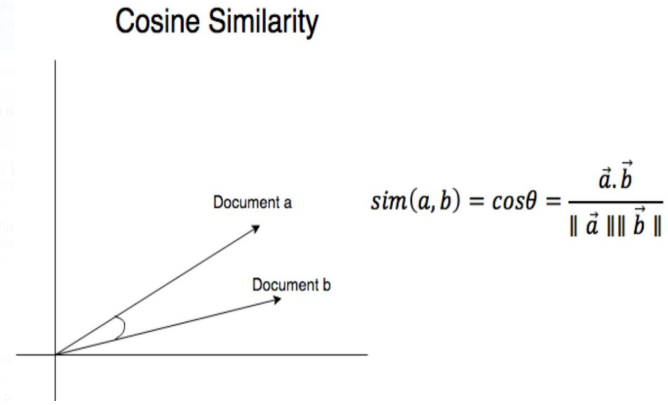
The basis for content-based filtering is the content of each an item. In our case the contents of the movie would be different attributes like genre, year of release etc. The profile of the user is built on the content of the item. If a user likes a particular item, the next time a recommender system recommends an item to this user, it will be similar to the one that the user has liked in the past. For example, a user likes a particular food item on a website which is sold out, the more items that will be recommended to this user will be the food products with similar ingredients in them. Here the 'item' is the food product and the 'content' are the ingredients.

The content from an item that a user likes builds the user profile and the content of an item itself builds the item profile. As the user explores more items, the user profile keeps on building and getting more accurate in finding the best matches to be recommended to the user.

TF (term frequency) and IDF (inverse document frequency) are the two important factors used for the information retrieval. TF can be explained as the frequency of a word in a text and IDF is the inverse of the text frequency in the collection of different texts. If we search "the features of normal distribution" on Google, it is certain that "the" will

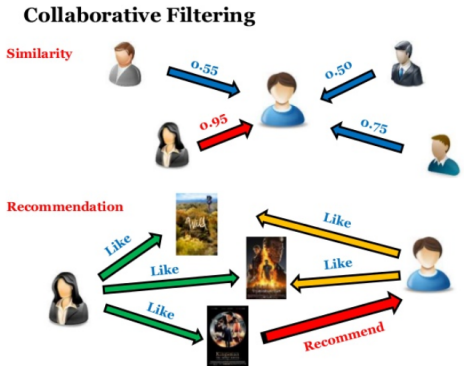
occur more frequently than “normal distribution”, but the importance of normal distribution is higher than the search query point of view. TF-IDF weighting negates the effect of high frequency words in determining the importance of an item.
In the movie recommender system, we will use ‘Genres’ as a significant content for recommending a movie and TF-IDF will contribute in finding similar movies by converting the data to vector space model. If we want to check similarity between two movies, one of which is an ‘action’ movie and other being a ‘drama’ movie, a vector will be built for each of the movies. The angle between these vectors is found out using the cosine similarity and the angle being smaller indicates two movies being more similar.

TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document).
IDF(t) = log_e (Total number of documents / Number of documents with term t in it).
TF(t) = (Number of occurence of t in a document) / (Total number of terms in the document).
IDF(t) = log_e (Total number of documents / Number of documents with term t in it).

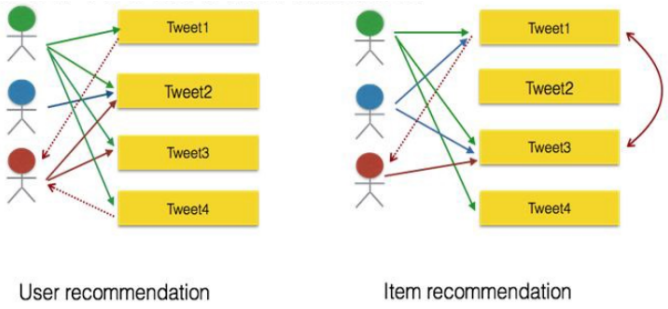


Collaborative Filtering:
Collaborative filtering uses a user’s interest and tries to find users with similar interest to recommend an item, in our case movies. Basically, it assumes that if user ‘A’ and ‘B’ have the same opinion about a movie, then both the users will have a similar opinion about another movie as well.
Different types of collaborative filtering are: -

1) User Based Collaborative Filtering
2) Item Based Collaborative Filtering



In the above figure we can see that the female in the left side and the and the male figure in the right side have the same liking for two movies as a result of which the third movie liked by the female is recommended to the male.
In user based collaborative filtering, similarity between users is computed. We are using cosine similarity as the metric here in our project. The explanation for the diagram mentioned above is what exactly user-based collaborative filtering is.
In item based collaborative filtering, similarities between different items in the dataset are calculated using one of the similarity measures and then these similarities are used to predict ratings for a user-item pair that is not present in the dataset.
The figure below shows the different between user-based and item-based collaborative filtering.



METHODOLOGY

In this project, the movielens database is used.
This dataset contains data for 3884 movies from 6041 users and the rating provided by these users for different movies.

Data pre-processing:
We checked how different attributes from the users tables plot against the ratings they done for movies. Movies were grouped according to the genre and the popularity of genres were computed based on the number of movies in a genre. It was found that

many movies in the dataset could fall under multiple genres, so the genres were separated with a '|' in the data frame.

As we have different data frames that were loaded with 3 different csv files (user, movies and ratings), we further merge all the data frames into a single data frame on which we perform our analysis.

Content-based Collaborative Filtering:

We start with performing content-based filtering on our data frame. The column 'genre' is then used to perform content-based collaborative filtering by passing the dataset into the vector space model using the TfidfVectorizer from the Scikit-learn library.

After we have the data into the vector space model, we find out the similarity between movies using the cosine similarity by linear_kernel in the Scikit-learn library.

We used a function to find similarity between a movie with other movies. This function takes the movies name as the input, finds the corresponding movie id and then computes scores to find other similar movies with a single movie (the movie input name that we use for the function). The result is sorted in descending order of similarity scores, the most similar movies being at the top.

Collaborative Filtering:

The computation is done on a small fraction of the dataset as the dataset is very large and the system might run out of memory for the computations that we perform.

Item-Item collaborative filtering:

A matrix is created which holds the ratings for movies that were provided by users for each movie. Using this matrix with the ratings, a new matrix is formed using 'pairwise distances' to see how similar a movie is with other movies. A value which is closer to a '1' indicates how similar a movie is.

A function is implemented for recommending movies to a user which. This function considers movies which a user has watched in past and gave 4.5 or 5 rating to these movies. These movies are matched with movies and similarity scores greater than 0.45 are considered for recommendation to the user. The movies that are finally recommended to the user are the movies which this user has never seen.

User-Item collaborative filtering:

A matrix is formed using the user id and movie id with ratings as the values against each user for each movie.

The similarity matrix is formed for user using the user-movie rating matrix, a value closer to '1' indicates that a user is completely similar to another user with which the score is '1'.

The function implemented here uses this similarity matrix and recommends movie by taking input as the 'userid' and recommending movies to that user. The list of movies are the ones that user has not seen and have been recommended by checking another user with which the similarity score is high.

Hybrid Model:

In the hybrid model we have used the LightFm implementation. The LightFm includes a loss function which tell us how well our model does in the predictions.

The LightFm datasets has the movielens dataset which we have imported to the notebook. We have extracted a dataset that has a minimum rating of 0.5. We create the model in lightfm with the warp loss function. The training data is set into the model with epoch of 30.

A function is implemented where we pass the model, data and userids. For each user, the movies they have liked is found out with the data we have imported. The predict method in lightfm helps us with this. This function when input with an array of userids will output the movie names which are the recommendations for that particular user.

K-Means Clustering:

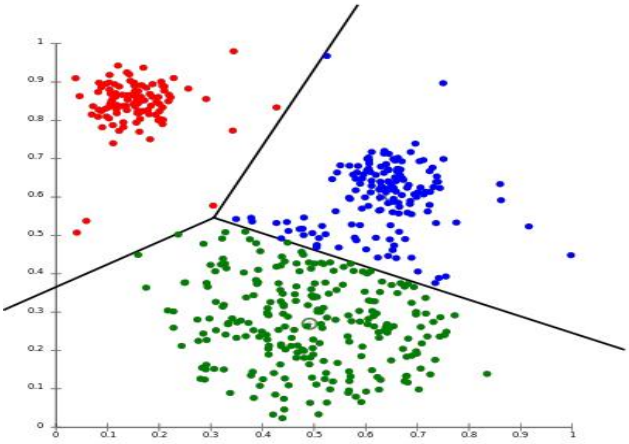
We have used k-means clustering here to explore how a user's taste in movies varies. The basis of this is the rating. This technique has helped us understand how ratings play a significant role in the recommender system. We have targeted user-item collaborative filtering and clustering together all the users having the same taste on the basis of movie ratings provided by them. If a user has not watched any movie, we can predict the rating of the movie as the average of that cluster on the basis of the ratings provided by other users in the same cluster. In k-mean clustering the hyper parameter is k number of clusters whose value is determined by plotting the elbow curve for different values of k in the given range.

Singular Value Decomposition: -

SVD is a matrix factorization technique where we reduce the dimensions of the sparse matrix by decomposing into two matrices i.e. user matrix and item matrix. The sparse matrix is denoted as the dot product of these two matrices. SVD doesn't work with matrices where there is null data, but we have replaced these values with '0' values. Sparse matrix is decomposed such that the squared error difference between the dot product of decomposed matrix and known rating in the sparse matrix is minimum. Regularization is performed by selecting a suitable value of lambda in order to avoid overfitting of data. The advantage of using SVD is that it avoids the Cold Start problem.

K-Mean Clustering: -

The algorithms that we have implemented to recommend movies to a user have their own disadvantages, cold start problem being one of them. For predicting a rating of a movie that a user has not seen in the past, user-item sparse matrix from collaborative filtering is used in clusters. Users who have same preferences are brought together in one cluster. Movies are segregated into their respective genre types.



The average rating given by a user to the movie belonging to a genre type is calculated. This calculated value is then the base of clustering down the matrix into k clusters and by analyzing an elbow curve against several values of k the optimum of k is being selected. The input is taken from a user about the user if, movie rating and movie genre, to query the movie data.

We start by getting the list of all unique genres in the dataset and the number of times they occurred in the dataset. This is followed by getting the average value for each of reach genre rating given by each user. The resultant data frame consists of NaN values which are replaced with the mean values of each category.

The elbow curve shows us what the better choices of k should be for the clustering. Any other value would return in bad clusters.

We further analyze how users rated individual movies. The resultant matrix contains a lot of NaN values as a lot of users have not rated or seen movies. For the research point of view, we take subset of complete dataset by selection the most rated movies and the users who have given maximum ratings. We can then create the sparse matrix that will give us a rating provided by a particular user to a particular movie. We can then map this matrix into the clusters where each cluster represents the users with similar taste/preference. So, benefit of using this approach is that even if a user has not watched the movie, we could still predict the rating of the movie based on the mean value of the cluster.

RESULT

Sr.No	Model	Accuracy
1	SVD	RMSE – 0.8740
2	LightFm	AUC – 0.90
3	K-Mean Clustering	Silhoutte 0.075

CONCLUSIONS

In this paper, we discussed the problems that are faced in recommender systems with algorithms and attempted a new algorithm to overcome the shortcomings of the existing algorithms. A user's preference should be one of the most important factors that should be considered in any recommendation system. We implemented the existing algorithms used in the movie recommendation system and how accurate they are making predictions. We then showed how we could take input from a user and used pre-filtering and k means clustering. We have tried to map the relationship of user & item using the k-means clustering approach. For a new user, we can recommend the movies of genre type with the highest average rating.

REFERENCES: -

1) https://towardsdatascience.com/the-4-recommendation-engines-that-can-predict-your-movie-tastes
2) https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/
3) http://files.grouplens.org/datasets/movielens/ml-100k-README.txt
4) https://beckernick.github.io/matrix-factorization-recommender/
5) https://medium.com/@james_aka_yale/the-4-recommendation-engines-that-can-predict-your-movie-tastes-bbec857b8223
6) https://hackernoon.com/introduction-to-recommender-system-part-1-collaborative-filtering-singular-value-decomposition-44c9659c5e75?gi=da97d0d6404