

Movie Recommendation Systems

CSN-382 Project

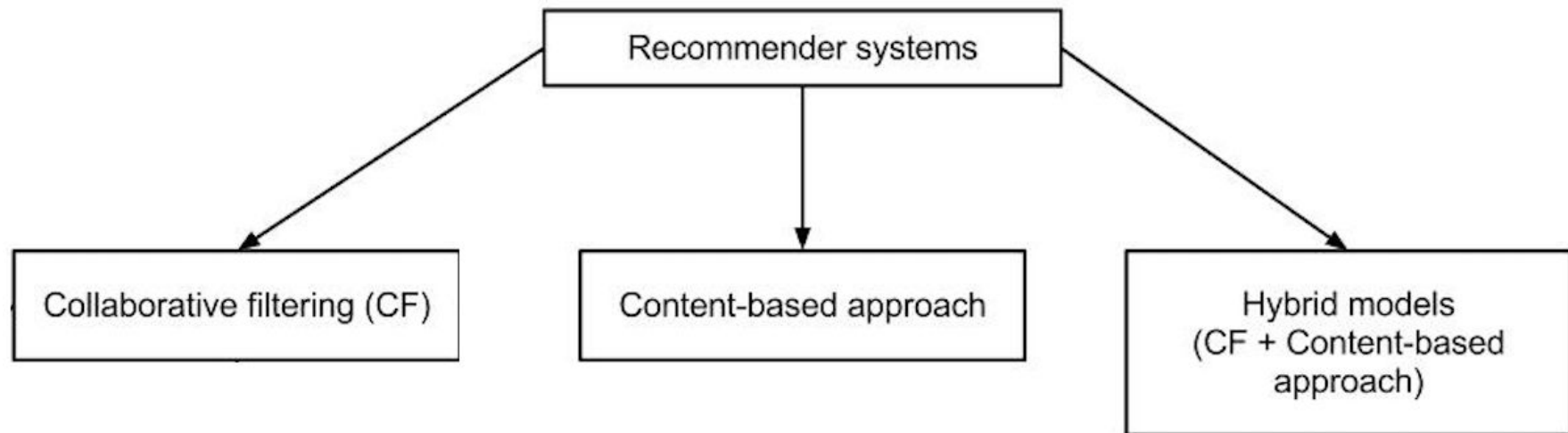
Submitted By:
Abhishek Jaisingh, 14114002
Tirth Patel, 14114036
Sahil Garg, 14114046
Sumit Kumar Singh, 14114063

Recommendation System

Recommendation systems produce a ranked list of items on which a user might be interested, in the context of his current choice of an item.

- ❖ Subclass of Information filtering system that seek to predict the 'rating' or 'preference' that a user would give to them.
- ❖ Helps deciding in what to wear, what to buy, what stocks to purchase etc.
- ❖ Applied in variety of applications like movies, books, research articles.

Recommendation systems has mainly two elements Item and User.



Dataset Usage

We have used MovieLens Dataset by GroupLens

This data set consists of:

- ❖ **100,000** ratings (1-5) from 943 users on 1682 movies.
- ❖ Each user has rated at least 20 movies.
- ❖ Simple demographic info for the users (age, gender, occupation)

Since we have developed a prototype of hybrid recommendation system. We have also **scraped** the **content-based** data from IMDB for the movies we already had for collaborative filtering purpose in the movielens dataset.

Movie Recommendation System

1. Content Based: The recommendation system recommends other movies which are similar to that selected movie.

$$f(\text{movie}) \rightarrow \{\text{movies}\}$$

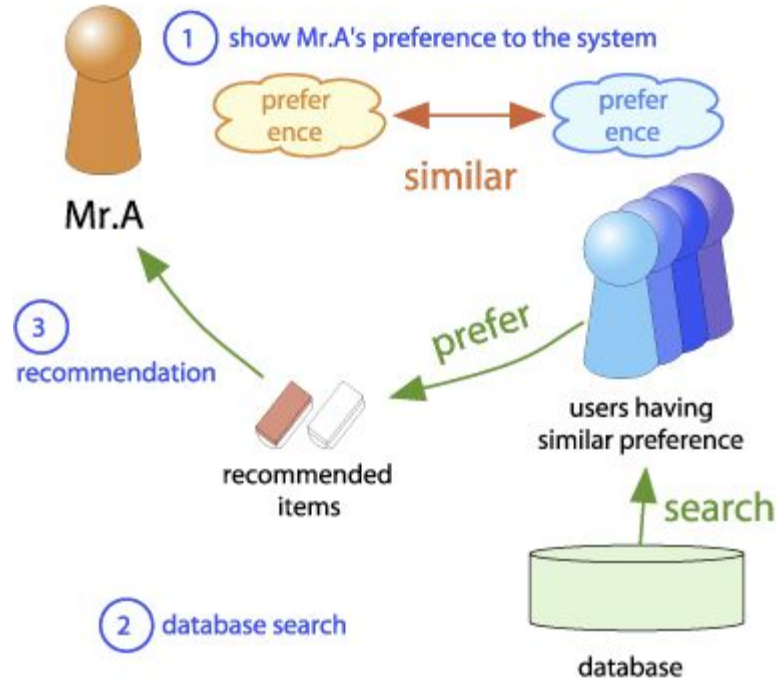
2. Collaborative: The recommendation system recommends movies which are rated highly by the similar users.

$$f(\text{movies}, \text{user}) \rightarrow \{\text{movies}\}$$

Collaborative Filtering

- ❖ Collaborative Filtering system maintains a database of many users' ratings of a variety of items.
- ❖ Makes use of the user data, ignoring content / item data.
- ❖ Almost all existing commercial recommenders use this approach (e.g. Amazon, Facebook, LinkedIn).

Basic Idea behind Collaborative Filtering




























Collaborative Filtering

Utility Matrix : Users have preferences for certain items and these preferences must be discovered from the data. The data is represented as a utility matrix, a value that represents the rating given by that user for that item and is given for each user-item pair.

The goal of the recommendation engine is to predict the blanks in a utility matrix.

Utility Matrix

Similarity Measures

Pearson Correlation Similarity Measure : Measure of similarity of users or items from the rows and columns of the Utility Matrix.

Advantages:

- Pearson Correlation Measure is easy to interpret.
- Tends to give better results than other similarity measures.
- Normalizes the ratings.

Other Similarity Measures:

Euclidean Distance, Manhattan Distance, Cosine Similarity

PCS Measure

Let $r_{x,i}$ denote the rating given by user x to item i . If I is the set of all items that two users x and y have both rated, then the Pearson Correlation Similarity Measure between the two users is given by:

$$pcs(x,y) = \frac{\sum_{i \in I} (r_{x,i} - \bar{r}_x)(r_{y,i} - \bar{r}_y)}{\sqrt{\sum_{i \in I} (r_{x,i} - \bar{r}_x)^2} \sqrt{\sum_{i \in I} (r_{y,i} - \bar{r}_y)^2}}$$

where \bar{r}_x denotes the average rating given by user x to all items. To calculate \bar{r}_x we only consider items that were rated by the user.

Prediction

- One way of predicting the value of the utility matrix entry (estimated rating) of a given user u for item i , is to average the ratings of top_ n users.
- Other approach is to first normalize the utility matrix.
- That is, for each of the n most similar users, subtract their average rating for all items from the rating of the item of interest i . Take the average of these differences for those users who have rated i , then add this average difference to the average rating that u gives for all items.

Results

- We achieved a Mean Square Error of 1.076 for the prediction of user ratings and $\text{top_n} = 150$ (neighborhood size).

Disadvantages

1. Cold Start: There needs to be enough other users already in the system to find a match.
2. Sparsity: Most users do not rate most items and hence the user-item matrix is typically very sparse. It is hard to find users that have rated the same items.
3. First Rater: It is not possible to recommend an item that has not been previously rated. This problem comes for new items mostly.
4. Popularity Bias: CF cannot recommend items to someone with unique tastes. In that case there is a tendency to recommend the popular.

Content Based Filtering

- ❖ It uses only the item data maintaining a profile for each item. Each user is assumed to operate independently. No need for data on other users.
- ❖ Considering the attributes or feature of the item, it finds the similarity between items, and recommends the most similar item for an item.
- ❖ If we consider the content of a movie as director, writer, cast etc., then each of these attribute can be considered as a feature.

Similarity

We recommend the items to the users which are very much similar to the rated item by the user.

We define similarity S between objects O_i and O_j as

$$S(O_i, O_j) = f(A_{1i}, A_{1j}) + f(A_{2i}, A_{2j}) + \dots + f(A_{ni}, A_{nj})$$

Here, $A_{1i}, A_{2i} \dots A_{ni}$ are the features for the item i .

Function $f(A_{1i}, A_{1j})$ represents the distance (similarity) between the 1st feature for item i and j .

Features and Distance Measures

<i>Feature</i>	<i>Type</i>	<i>Domain</i>	<i>Distance Measure</i>
Release	Year	YYYY	$\frac{(300 - Y_1 - Y_2)}{300}$
Type	String	Movie,TV,VG,V,mini	$T_1 = T_2 ? 1 : 0$
Rating	Integer	(0-10)	$\frac{(10 - R_1 - R_2)}{10}$

Features Used in Movie Recommendation
with their distance measures

Disadvantages

1. Cannot filter items on some assessment of quality, style or viewpoint because of lack of consideration of other people's experience.
2. Absence of personal recommendations.
3. No serendipitous items i.e. the ability of the system to give an item surprisingly interesting to a user, but not expected or possibly foreseen by the user.

Hybrid Approach

We attempt to hybridize collaborative filtering and content based recommendation. Item similarity measure used in content based recommendation is learned from a collaborative social network of users.

In content based recommendation every item is represented by a feature vector. The features hold numeric or nominal values representing certain aspects of the item.

A variety of distance measures between the feature vectors may be used to compute the similarity of two items.

Content Based Similarity

Users base their judgments on some latent criteria which is a weighted linear combination of the differences in individual attribute.

Accordingly, we define similarity S between objects O_i and O_j as

$$S(O_i, O_j) = \omega_1 f(A_{1i}, A_{1j}) + \omega_2 f(A_{2i}, A_{2j}) + \dots + \omega_n f(A_{ni}, A_{nj})$$

where ω_n is the weight given to the difference in value of attribute A_n between objects O_i and O_j , the difference given by $f(A_{ni}, A_{nj})$.

weights $\omega_1, \omega_2, \dots, \omega_n$ are obtained from a social collaborative network.

Weights From Collaborative Network

We describe below a linear regression framework for determining the optimal feature weights.

The edge weight between vertices O_i and O_j ,

$$E(O_i, O_j) = \# \text{ of users who are interested in both } O_i, O_j$$

This may be considered as human judgment of similarity between O_i, O_j .

$$\omega_0 + \omega_1 f(A_{1i}, A_{1j}) + \omega_2 f(A_{2i}, A_{2j}) + \cdots + \omega_n f(A_{ni}, A_{nj}) = E(O_i, O_j)$$

Solving the above regression equations provide estimates for the values of $\omega_1, \omega_2, \cdots, \omega_n$. If there are I movies under consideration, it is possible to have IC^2 regression equations of the above form.

Prediction

- Using regression we can solve for the weight vector, W
- User can input the movie for which he wants recommendation (say O_i)
- We check similarity, $S(O_i, O_j)$ of the given movie with all other movies (O_j).
- Each movie's similarity score is $\text{dot_product}(S, W)$.
- We have to recommend movies which have the maximum similarity score

Future Work

- In collaborative filtering, we have a problem of sparsity of data. Very few users actually rate the same movie.
- We can use Clustering Algorithms like K-Means to cluster items or users or both based on their attributes.
- In the hybrid approach, we can use more features to get better predictions. (Currently, we have only 9 features)

References

<https://grouplens.org/datasets/movielens/100k/> - MovieLens Dataset.

<https://pdfs.semanticscholar.org/1356/f4eda338b58b2840c5f643a988a1008806f0.pdf> - Machine Learning Based Hybrid Recommendation System

Thanks