MACHINE LEARNING PROJECT

MOVIE RECOMMENDER

Department of Computer Science
University of Delhi

Submitted by :
Sumit(59)
Deepali(15)

Submitted to:
Dr. Bharti Rana

1. Background and problem definition

In the modern world, there is an endless variety of content consumed such as books, videos, articles, movies, etc., it is very difficult for users to find content they are really interested in. And for the digital content provider, they want to engage more and more users in their service for maximum time. That is why many researchers and companies develop Recommender Systems. The aim of the recommender system is to connect users and information, which in one way helps users to find information valuable to them and in another way pushes the information to specific users.
Providing related content out of relevant and irrelevant collection of items. Although the traditional recommendation systems get the job done, these recommender systems have cold start, sparsity and scalability issues. Also most traditional systems either use collaborative filtering or content based filtering which have their own disadvantages.Content based filtering cannot provide new unseen recommendations while collaborative filtering has issues such as cold start, sparsity.

Cold start: For a new user or item, there isn't enough data to make accurate recommendations.
Scalability: In many of the environments in which these systems make recommendations, there are millions of users and movies. Thus, a large amount of computation power is often necessary to calculate recommendations
Sparsity: The number of movies listed on content provider sites is extremely large. The most active users will only have viewed/rated a small subset of the overall database. Thus, even the most popular movies have very few views/ratings.

Netflix	2/3rd of movies watched are recommended
Google News	Recommendation generated 38% more click-throughs
Amazon	35% more sales from recommendations
Choice stream	28% people would buy more music if they found what they liked

Companies benefit through recommendation system

2. Related work

2.1. Paper - 1

The proposed algorithm for the recommendation system is collaborative filtering algorithm. Their algorithm worked on Singular Value Decomposition and cosine similarity and predicted a list of

top n products recommended to the user. The SVD can handle the massive dataset and sparseness of the rating matrix. A cosine similarity algorithm is used to measure the similarity between users. Those users rated the same movie have similar choices or preferences. User-based collaborative filtering systems have been very successful in the past, but their widespread use has revealed some potential problems such as scalability and cold start.

Step 1: Created users-items matrix, 0 if a movie is not rated by user show in Fig.1 .

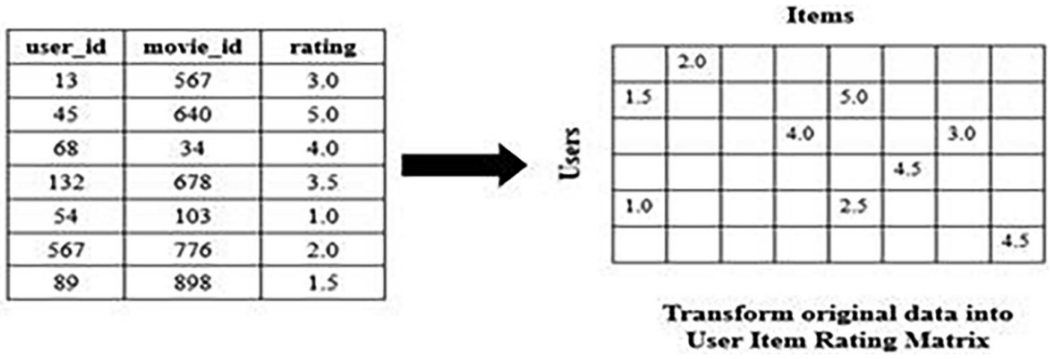


Fig.1

Step 2 : Used SVD to reduce the feature of dataset and space dimension from N dimension to K dimension (where K < N)

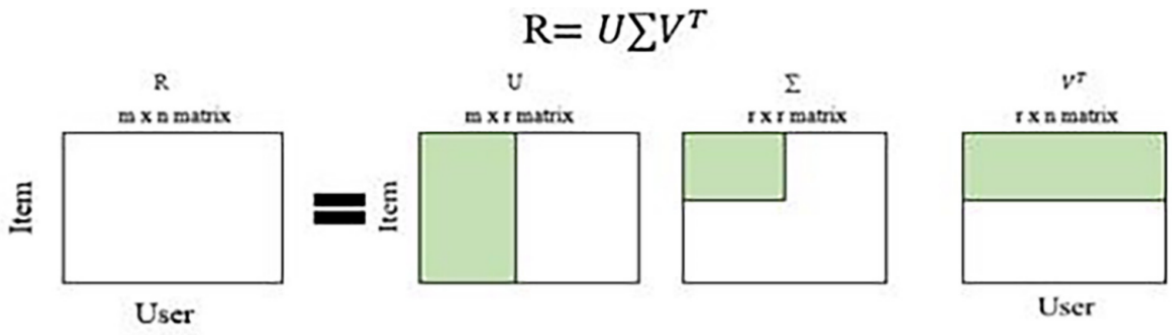


Fig. 2

Step 3 : Used cosine similarity to find k similar users.

$$Cos\,\theta = \frac{\vec{a}, \vec{b}}{\|\vec{a}\|\|\vec{b}\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2}\sqrt{\sum_1^n b_i^2}}$$

Step 4 : Computed rating for movie i, for user u and v is a set of K similar users.

$$P(u,i) = \bar{R}_u + \frac{\sum v \in K\, sim(u,v)\left(r_{v_i} - r_v\right)}{\sum v \in K|sim(u,v)|}$$

2.2. PAPER - 2

The proposed algorithm for the recommendation system is content-based filtering. Their approach first recommends similar movies based on the user's choice and thereafter performs sentiment analysis on the reviews of the movie chosen. Although the system is accurate, it does have some limitations. Since it is a content-based algorithm therefore it only recommends items that are similar to only those that a user has liked/viewed in the past. One more limitation is the linguistic barrier while doing the sentimental analysis. As of now only reviews written in English can be analyzed using their algorithm. Moreover sentiment analysis gives wrong classification if the reviews are sarcastic or ironic.

2.2.1. Movie Recommendation

Took genre, keywords,crew and cast columns from the dataset and combined them to form a single column Tags. Thereafter applied count vectorizer on tags and then applied cosine similarity. With the score of cosine similarity generated list of tok k recommended movies.

2.2.1.1.Count vectorizer

It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. CountVectorizer creates a matrix in which each unique word is represented by a column of the matrix, and each text sample from the document is a row in the matrix. The value of each cell is nothing but the count of the word in that particular text sample

2.2.2. Sentiment Analysis

Sentiment analysis on recommended movies is done using SVM and Naive bayes.

2.2.2.1. Naive Bayes

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

where A and B are events and P(B) ≠ 0.

2.2.2.2. SVM

Radial Basis Function Kernel was used which is a type of Non-linear SVM . It is referred to as the RBF kernel. Metric squared Euclidean distance is used for distance. It is used to draw completely non-linear hyperplanes.

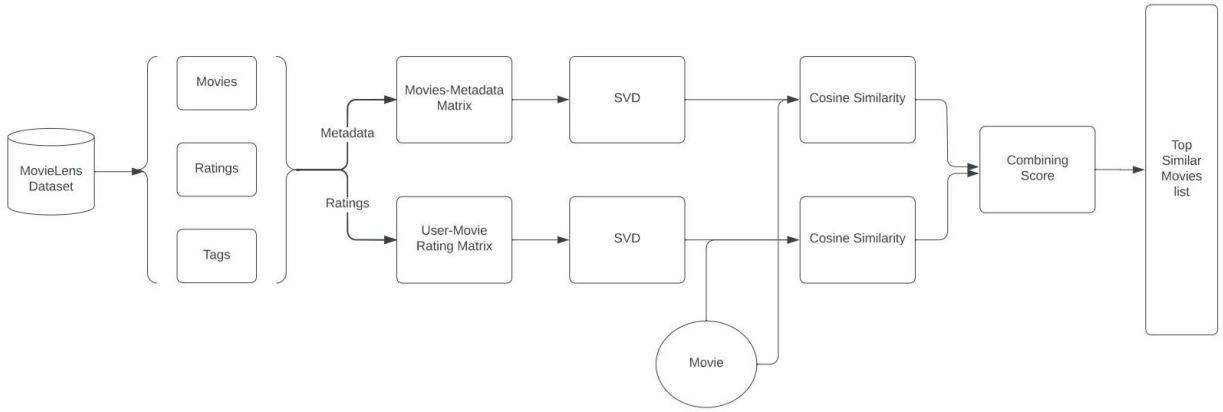
$$K(X, X') = exp\left(-\frac{\|X - X'\|}{2\sigma^2}\right)$$

Where ,

X and X' are two points
' σ' is the variance and the hyperparameter.
||X –X'|| is the Euclidean (L 2 -norm) Distance between two points X and X'.

3. Methodology

We proposed a hybrid algorithm for the movie recommendation system. The algorithm uses both content based and collaborative filtering algorithms. We predict top k movies similar to a given movie. We used SVD to reduce the dimension of the matrix for faster computation. To find similarity between two movies cosine similarity is used. Unlike other algorithms which are using only content based or collaborative filtering we are using hybrid to overcome their limitations and use advantages of both.



Architecture of proposed recommendation system

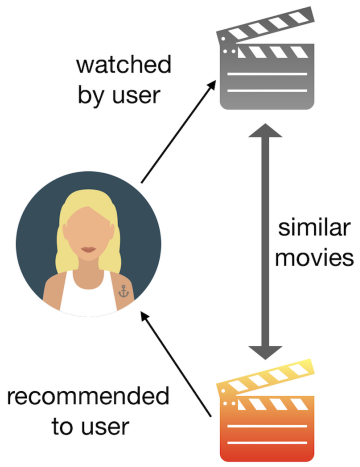
3.1. Algorithm Steps

1. Create metadata by combining movie genres and tags.
2. Build movie-metadata matrix for content based approach.
3. Reduce dimensions of movie-metadata matrix using SVD.
4. Build user-movie rating matrix for collaborative filtering approach, we will mark 0 if user has not rated a movie.
5. Reduce dimensions of user-movie rating matrix using SVD.
6. For a given movie apply cosine similarity and find how much other movies are similar to the given movie.
7. Find average scores from both models (content and collaborative) and sort the scores in descending order.
8. List out top k movies from sorted list.

3.2. Content based

Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback. The system finds the similarity between products based on its description. The user's previous history is taken into account to find similar products the user may like.

For example, if a user likes movies such as 'Mission Impossible' then we can recommend to him the movies of 'Tom Cruise' or movies with the genre 'Action'.



Source : google images

Advantages

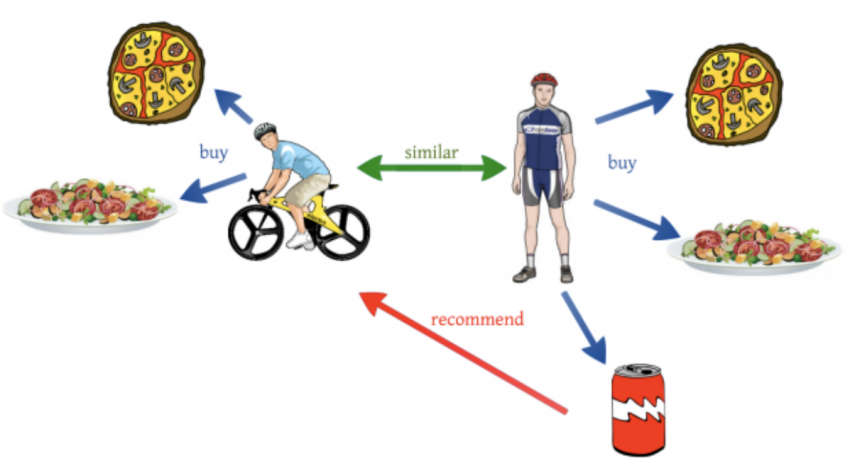
- The model doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to a large number of users.
- The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in.

Disadvantages

- Since the feature representation of the items are hand-engineered to some extent, this technique requires a lot of domain knowledge. Therefore, the model can only be as good as the hand-engineered features.
- The model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

3.3. Collaborative

In Collaborative Filtering, we tend to find similar users and recommend what similar users like. For example, if the user ‘A’ likes ‘Coldplay’, ‘The Linkin Park’ and ‘Britney Spears’ while the user ‘B’ likes ‘Coldplay’, ‘The Linkin Park’ and ‘Taylor Swift’ then they have similar interests. So, there is a huge probability that the user ‘A’ would like ‘Taylor Swift’ and the user ‘B’ would like ‘Britney Spears’. This is the way collaborative filtering is done.



Source : google images

Advantages

- New products can be introduced to the user.
- Business can be expanded and can popularize new products.

Disadvantages

- The new item cannot be recommended if no user has purchased or rated it.

3.4. SVD

The Singular Value Decomposition (SVD), a method from linear algebra that has been generally used as a dimensionality reduction technique in machine learning. SVD is a matrix factorisation technique, which reduces the number of features of a dataset by reducing the space dimension from N-dimension to K-dimension (where K<N). It uses a matrix structure where each row represents a user, and each column represents an item(movie). The intuition behind matrix factorization is fairly simple, given some user-item matrix, you want to decompose that matrix such that you have a user matrix and an item matrix independently.

3.5. Cosine similarity

Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction.
The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.

When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

$$Cos\,\theta = \frac{\vec{a}, \vec{b}}{\|\vec{a}\| \left\|\vec{b}\right\|} = \frac{\sum_1^n a_i b_i}{\sqrt{\sum_1^n a_i^2} \sqrt{\sum_1^n b_i^2}}$$

4 Experiment

4.1. Dataset

We have used the dataset of MovieLens which is publicly available to conduct experiments. There are two versions, full and small.
Full: 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users. Includes tag genome data with 14 million relevance scores across 1,100 tags.
Small: 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users.
We have used three csv files i.e, movies.csv, ratings.csv, tags.csv from the small dataset.

4.2. Experimental platform

We have implemented our experiment in Python 3.10.8 using the Jupyter notebook on a windows machine having intel core i5 8th Gen and 8Gb RAM.

4.3. Procedure

4.3.1. Pre Processing

We have done feature engineering to remove unwanted columns from our dataset which are not required.
We have concatenated genres and tags columns into one single column per movie named as metadata. We used TF-IDF to create a matrix of movies-metadata.We have also created a user-movie rating matrix where each user has a rating to all movies. 0 if rating for a particular user-movie pair is not present.

4.3.1.1. TF-IDF Term Frequency Inverse Document Frequency

It can be defined as the calculation of how relevant a word in a series or corpus is to a text. Term frequency is defined as the number of times a word (i) appears in a document (j) divided

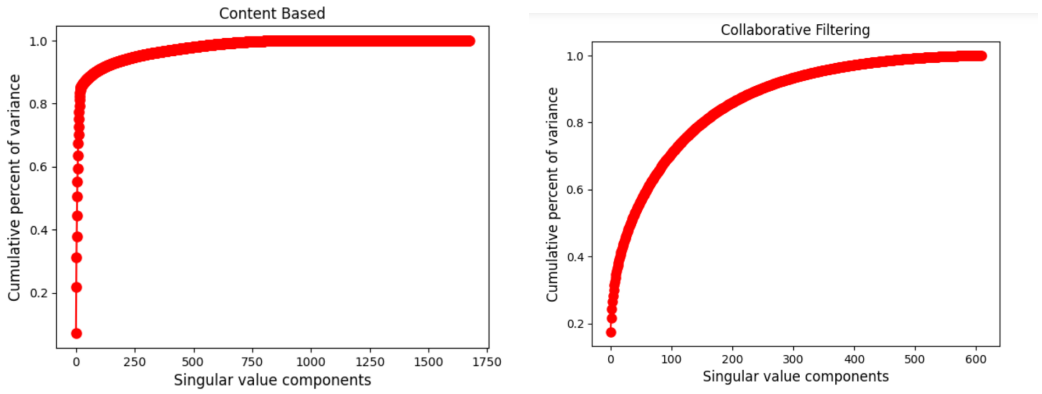
by the total number of words in the document.Inverse document frequency refers to the log of the total number of documents divided by the number of documents that contain the word.

$$W_{x,y} = tf_{x,y} * \log(N/df_x)$$

where,
$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

4.3.2. SVD

We have applied SVD to reduce dimensions and to keep latent features that explains maximum variance. Certain features are not directly observable, but they are necessary for rating predictions. The set of hidden features are called Latent features. We don't know specifically what each latent feature represents. Still, one feature can be assumed to represent that a user likes a comedy film, and another latent feature may reflect that the User likes animation movies and so on.



We can notice that 25 latent features in content based explains more than 80% of variance and 120 latent features in collaborative filtering can explain nearly 80% of variance.

4.3.3 .Cosine Similarity

Finally we take a movie and find the scores of all movies using cosine similarity between this movie and other movies.
We do cosine similarity for both content based as well as collaborative filtering and take average of their result and sort the result in decreasing order and display top k movies.

4.4. Comparison

We have chosen a movie - Toy Story (1995) as input.

4.4.1. Content Based

We can see that Toy Story is an animated movie so the recommended movies with content based filtering are also biased towards animated children movies.

	content	collaborative	hybrid
Shrek the Third (2007)	0.999988	0.254639	0.627314
Moana (2016)	0.999988	0.064746	0.532367
Monsters, Inc. (2001)	0.999988	0.666134	0.833061
Asterix and the Vikings (Astérix et les Vikings) (2006)	0.999988	0.066157	0.533073
Tale of Despereaux, The (2008)	0.999988	0.107232	0.553610
Adventures of Rocky and Bullwinkle, The (2000)	0.999988	0.238890	0.619439
Emperor's New Groove, The (2000)	0.999988	0.354607	0.677298
Antz (1998)	0.999988	0.493237	0.746613
The Good Dinosaur (2015)	0.999988	0.170961	0.585475
Wild, The (2006)	0.999988	0.097743	0.548866
Turbo (2013)	0.999988	0.107245	0.553617
Shrek Forever After (a.k.a. Shrek: The Final Chapter) (2010)	0.999452	0.212961	0.606206
Ant Bully, The (2006)	0.999452	0.147837	0.573644
Toy Story 3 (2010)	0.999452	0.552711	0.776082
Inside Out (2015)	0.970594	0.400551	0.685572

4.4.2. Collaborative Filtering

Here movies other than animated movies are also shown.

	content	collaborative	hybrid
Jurassic Park (1993)	0.187327	0.766551	0.476939
Forrest Gump (1994)	0.091474	0.738866	0.415170
Apollo 13 (1995)	0.401606	0.730387	0.565997
Toy Story 2 (1999)	0.943806	0.725854	0.834830
Shrek (2001)	0.932965	0.719614	0.826289
Star Wars: Episode IV - A New Hope (1977)	0.105561	0.710716	0.408139
Raiders of the Lost Ark (Indiana Jones and the Raiders of the Lost Ark) (1981)	0.380470	0.704752	0.542611
Aladdin (1992)	0.720221	0.699505	0.709863
Star Wars: Episode VI - Return of the Jedi (1983)	0.201955	0.699347	0.450651
Groundhog Day (1993)	0.449217	0.695069	0.572143
Independence Day (a.k.a. ID4) (1996)	0.185611	0.694090	0.439851
Shawshank Redemption, The (1994)	0.002139	0.689865	0.346002
Pulp Fiction (1994)	0.109972	0.689672	0.399822
Lion King, The (1994)	0.666150	0.689513	0.677832
Men in Black (a.k.a. MIB) (1997)	0.087810	0.685108	0.386459

4.4.3 .Hybrid

Here we have a good mix of both algorithms i.e, content based and collaborative filtering.

	content	collaborative	hybrid
Toy Story 2 (1999)	0.943806	0.725854	0.834830
Monsters, Inc. (2001)	0.999988	0.666134	0.833061
Shrek (2001)	0.932965	0.719614	0.826289
Toy Story 3 (2010)	0.999452	0.552711	0.776082
Bug's Life, A (1998)	0.877190	0.656804	0.766997
Finding Nemo (2003)	0.877733	0.637458	0.757595
Antz (1998)	0.999988	0.493237	0.746613
Aladdin (1992)	0.720221	0.699505	0.709863
Ice Age (2002)	0.877138	0.541202	0.709170
Incredibles, The (2004)	0.795734	0.616061	0.705897
Who Framed Roger Rabbit? (1988)	0.820183	0.562389	0.691286
Inside Out (2015)	0.970594	0.400551	0.685572
Harry Potter and the Sorcerer's Stone (a.k.a. Harry Potter and the Philosopher's Stone) (2001)	0.809080	0.559542	0.684311
Lion King, The (1994)	0.666150	0.689513	0.677832
Emperor's New Groove, The (2000)	0.999988	0.354607	0.677298

5. Conclusions and Future Work

Traditional recommender systems used content based or collaborative filtering to do recommendations. However both of them have some limitations such as content based filtering only makes recommendations based on past behavior while collaborative filtering overcomes this issue but it has limitations like cold start, sparsity etc.
We proposed a hybrid approach using both content based filtering as well as collaborative filtering. We solved the problem of scalability and sparsity using SVD by reducing dimensions of the matrix. Further we used cosine similarity to find movies similar to a given movie.
Future work can be extended by evaluating the proposed algorithm.

6. References

https://www.sciencedirect.com/science/article/pii/S2214785321003242

https://www.sciencedirect.com/science/article/pii/S2666285X22000176

https://www.sciencedirect.com/science/article/pii/S0957417422016293

https://www.inf.ufrgs.br/bdi/wp-content/uploads/MastersDissertationRobertoTorres.pdf

https://developers.google.com/machine-learning/recommendation

https://www.geeksforgeeks.org/user-based-collaborative-filtering/?ref=rp

https://towardsdatascience.com/recommendation-system-matrix-factorization-svd-explained-c9a50d93e488

https://en.wikipedia.org/wiki/Singular_value_decomposition

https://en.wikipedia.org/wiki/Cosine_similarity

https://www.geeksforgeeks.org/cosine-similarity/

7. Contribution of each member in various sub-sections

Sumit : Section - 1, 2.1, 3.1, 3.3, 3.4, 3.5, 4
Deepali : Section - 1, 2.2, 3.1, 3.2, 3.4, 3.5, 4