Movie Recommendation System

S. Rajarajeswari, Sharat Naik, Shagun Srikant, M. K. Sai Prakash and Prarthana Uday

Abstract Recommender systems are called information filtering tools, which use big data to recommend likes of the user according to their preference and interest. Moreover, they also help in matching users with similar tastes and interests. Due to this, a central part of websites and e-commerce applications is taken up by the recommender systems. Systems using recommendation algorithms like Collaborative Filtering, Content-Based Filtering, etc., are called Recommendation systems. Recommendation systems are transforming the way quiescent websites corresponds with their users. Rather than providing an orthodox experience in which users search for the products which they want and potentially buy products, recommender systems increase communication to provide a better experience. The work would be to implement both collaborative as well as content-based recommenders available and try to extend the knowledge obtained to a more efficient hybrid model. Then benchmark these hybrid algorithms for accuracy and computation time as well.

Keywords Recommendation systems · Collaborative · Cosine similarity · SVD · Machine learning · Python · Surprise library

1 Introduction

It is expected that in future with the increased use of computers, the use of advanced and latest technologies by users and professionals and decision-makers will increase a lot. The basic idea of recommendation systems is that, if some users share the same interests, or maybe in the past, e.g., they liked the same book, they might also have similar tastes in the future. Most recommender systems take three basic approaches: simple recommender, collaborative filtering, and content-based filtering. A hybrid approach can combine these approaches to come up with a more optimal solution.

A recommendation system has become an indispensable component in various e-commerce applications. Recommender systems collect information about the user's

S. Rajarajeswari (✉) · S. Naik · S. Srikant · M. K. Sai Prakash · P. Uday
Department of Computer Science Engineering, RIT, Bangalore 560054, India
e-mail: raji@msrit.edu

N. R. Shetty et al. (eds.), *Emerging Research in Computing, Information, Communication and Applications*, Advances in Intelligent Systems and Computing 882,
https://doi.org/10.1007/978-981-13-5953-8_28

preferences of different items (e.g., movies, tourism, TV, shopping,) by two ways, either implicitly or explicitly.

A movie recommendation system ideally analyzes factors like review, cast, plot, crew, genre, popularity and comes up with the best possible result. It takes into account a user preference and hence provides results based on the user's personal taste and choice. By increasing efficiency and accuracy of recommender systems, users expect lightning fast results and different types of recommender systems available today provide the same. There are four different kinds of recommendation systems, improving the accuracy and user personalization with each.

The four systems are the following:

Simple Recommendation System: A simple recommender is the most basic recommendation system available to us. It usually compares a metric/weight for all the available entities and provides the recommendation on the basis of this comparison.

Content-Based Recommendation System: This recommendation system involves the use of more attributes from the dataset to provide a well-filtered recommendation to the user. In this project, at first, the recommender just provides a recommendation using the movie overview and taglines. But there is a possibility that a user watches a certain movie for the director or the cast rather than the ratings. For this reason, the recommender takes into consideration the director, cast, genre, and keywords to provide a better search result. This algorithm makes use of k-means algorithm through cosine similarity to find how similar two movies are and hence can be recommended together.

Collaborative Recommendation System: Both the recommendation systems mentioned above are efficient but they are not personalized. They would show the same results to all the users. But that should not be the real case. Movies should be recommended based on the user's personal taste. That is where collaborative filtering comes into the picture.

Here, the movie ratings provided by a particular user and using the SVD surprise library functions, evaluate the RMSE and MAE values. Since these values are comparable, a predict function is used that provides an estimate for a movie for that particular user using his ratings. This way, the movie recommendations provided to each user would be different depending on their tastes.

Hybrid Recommendation System: Having demonstrated all the three available recommendation models, a hybrid system is developed which uses the concepts of both content base as well as collaborative recommendation systems. Here, the user's ID and movie title are taken as the inputs. Using the movie title, 30 similar movies are collected in a database based on the title, cast, crew, overview, etc., using the cosine similarity and linear kernel concept. And then using the user's ID, his ratings are procured which is then passed as an argument to the svd.predict(). Through the list of those 30 movies, the ones most suited to the user's taste are recommended.

2 Related Work

In a Movie Recommender System called "MOVREC" by Manoj Kumar, D. K. Yadav, Vijay Kr Gupta.

It uses collaborative and content-based filtering using the k-means algorithm. This asks a user to select his own choices from a given set of attributes and then recommends him a list of based on the cumulative weight of different attributes using k-means algorithm [1].

The disadvantage is that it does not use large datasets, hence there will not be meaningful results.

In a paper by Rahul Kataria and Om Prakash named An effective movie recommender system which uses cuckoo search, a bio-inspired algorithm such as cuckoo search has exclusive background sensing abilities and employs a special method to facilitate the evolution of continuing resolutions into novel and quality recommendations by generating clusters with reduced time [2].

It has a limitation where if the initial partition does not turn out well then the efficiency may decrease.

In 2007 Weng, Lin, and Chen performed an evaluation study which says using multidimensional analysis and additional customer's profile increases the recommendation quality. Weng used MD recommendation model (multidimensional recommendation model) for this purpose. Multidimensional recommendation model was proposed by Tuzhilin and Adomavicius [3].

2.1 Sajal Halder: Movie Recommendation Using Movie Swarm

Movies swarm mining that mines a set of movies, which are suitable for producers and directors for planning new movie and for new item recommendation. Popular movie mining can be used to solve new user problems. It has the capability of handling both new users and new items. It is better than content-based and collaborative approaches when it comes to new users. This method, however, has a drawback of finding a group of users depending on the genre [4].

Yueshen Xu: Collaborative recommendation with User-Generated Content (UGC).

It proposes a UGC-based collaborative recommendation which integrates probabilistic matrix factorization and collaborative regression models. It reaches a higher prediction accuracy than most of the available models. It is efficient when it comes to cold start problem. The cost per iteration is more as compared to traditional CTR models [5] (Table 1).

Table 1 Comparative study related works and algorithms

S. No.	Technique	Algorithm	Drawback
1	Collaborative and content based	K-means	It is not suited for large datasets
2	Cuckoo search	Clustering	Efficiency decreases if the initial partition is not proper
3	Movie swarm	Mining	Drawback of finding a group of users based on the genre
4	User-generated content (UGC)	Regression analysis	Cost per iteration is more when compared to traditional models

3 Design

See Table 2.

General Architecture Diagrams

See Fig. 1.

Simple Recommender System

In Fig.2 Simple recommender gives a prediction based on the weighted rating (WR) calculated. It takes in vote_ count and vote_average from Movies_metadata.csv.

The TMDB Ratings is used to come up with our Top Movies Chart. IMDB's weighted rating formula is used

Mathematically, it is represented as follows:

$$\text{Weighted Rating (WR)} = (v/v + m.R) + (m/v + m.C)$$

Table 2 Description of attributes and dataset used in the project

Dataset name (.csv)	Description
Movies_metadata	Contains information about 45,000 movies featured in TMDB
Keywords	Contains movie plot keywords available in the form of a stringified JSON object
Credits	Contains cast and crew information, also saved as a stringified JSON object
Links	Contains TMDB and IMDB Ids of all the movies
Links_small	Contains a subset of links database with 9000 movie entries
Ratings_small	Contains a subset of 100,000 ratings from 700 users on 9000 movies

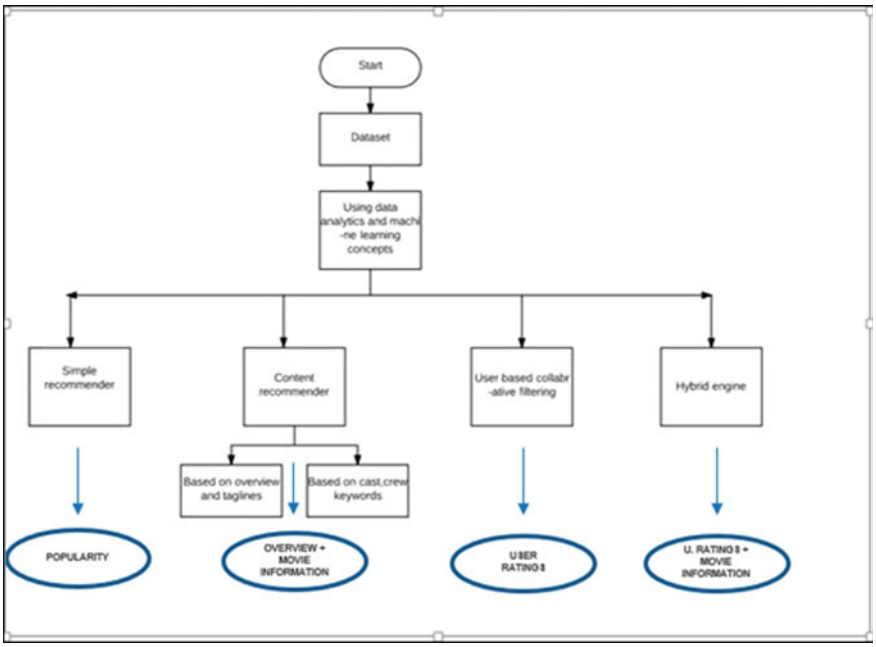

Fig. 1 General architecture of the recommendation system

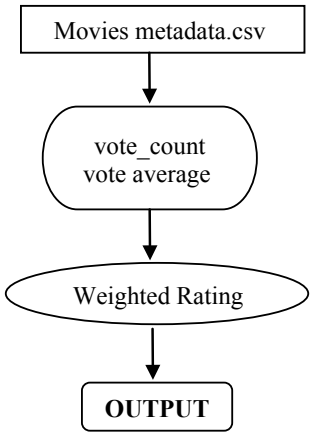

Fig. 2 Simple recommender

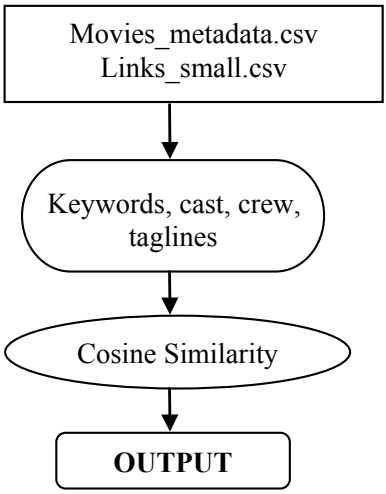

Fig. 3 Content-based recommender

Content Based Recommender System

In Fig.3, Content-based recommender gives prediction based on cosine similarity. It takes in keywords, taglines from Movies_metadata.csv & links_small.csv, it also takes director and cast as attributes.

To personalize our recommendations more, an engine is built that computes the similarity between movies based on certain metrics and suggests movies that are most similar to a particular movie that a user liked previously. Since movie metadata (or content) is used to build the engine, this is also known as Content-Based Filtering.

Cosine similarity is used to calculate a numeric value, which denotes the similarity between two movies.

Collaborative Recommender System

In Fig.4, Collaborative recommender gives prediction using singular value decomposition (svd). It takes in userID and movieID from ratings_small.csv. This is more personalized as it gives different suggestions to different users based on their taste.

A technique called Collaborative Filtering is used to make recommendations to People. Collaborative Filtering is based on the idea that users similar to me can be used to predict how much I will like a particular product or service those users have used/experienced but I have not.

Surprise library is being used that uses extremely powerful algorithms like Singular Value Decomposition (SVD) to minimize RMSE (Root Mean Square Error) and gives great recommendations to users.

Mean value is obtained for Root Mean Square Error, then we train our dataset and arrive at predictions.

Hybrid Recommender System

In Fig.5, Hybrid recommender gives a prediction based on the svd and cosine similarity. It uses Movies_metadata.csv as well as ratings.csv.

In this, content and collaborative filtering techniques are brought together.

Input given will be User ID and Title of a movie the user likes.

Output will be similar movies sorted on the basis of expected ratings by that particular user which is more personalized.

Pseudocode:

a. **Simple recommendation system**

build_chart(genre)
Input: genre to be searched for.
Output: Top 250 recommendation

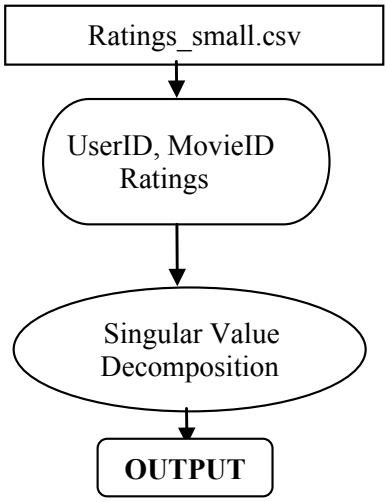

Fig. 4 Collaborative recommender

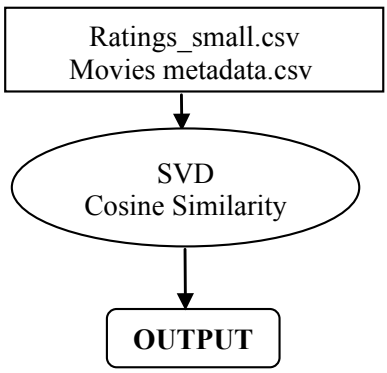

Fig. 5 Hybrid recommender

df= movies of 'genre'
vote_counts= number of votes for the genre
vote_average= average votes for the genre
C=vote_averages.mean()
m=vote_counts.quantile(percentile)
v=x['vote_count']
R=x['vote_average']
qualified['wr']= (v/(v+m) * R) + (m/(m+v) * C)
return qualified

b. **Content-Based recommendation system**

get_recommendations(title)
Input: Name of a movie
Output: Top 30 movie recommendation

Idx = id of the title
keywords= get keywords. Strip them of spaces.
Director= get_director(crew){if (job=='director') return name}
Cast= first three names in the cast
Count-_matrix=Get the count vectorizer matrix to calculate the cosine similarity using keywords, director, cast.
Cosine_similarity=Get the cosine similarity.
Return result.

c. **Collaborative Recommendation System**

recommendations(userId, title)
Input: User's ID and title of the movie
Output: Movies which cater to user's preference

Data = Use user id and movie id to get this
svd = Calling the SVD function.
Evaluate RMSE and MAE
Train data set to arrive at function
svd.train(trainset)
Predict using SVD and return the movies.

Proposed System

The proposed hybrid system takes features from both collaborative as well as content-based recommendation to come up with a system, which is more personalized for users.

Technique

Cosine similarity technique is used to come up with movies similar to the one given in the search engine. The selected 30 movies are then compared using the user's ratings and the SVD. Hence, the search advances to predict the movies most suitable for the user.

Algorithm:

The algorithm used for the hybrid engine is as follows:
def hybrid(userId, title):

Obtain the movieID (Idx) based on the title.
Obtain the sim_scores using cosine_similarity on the procured Idx
Use cosine similarity to obtain 30 movies similar to Idx using 'vote_count', 'year', 'vote_average' etc.
Using svd.predict(), calculate the RMSE between the suggested movies as well as the movie passed as an argument.
Sort the list obtained.
Return the top 10 movies of the list.

4 Results

Content and Collaborative filtering techniques are brought together to build an engine that gives movie suggestions to a particular user based on the estimated ratings that it had internally calculated for that user. It makes use of cosine similarity concept to find similar movies and the SVD prediction to estimate the movie recommendation as per the user's taste.

Simple Recommendation System:
See Fig. 6.
Content-Based Recommendation System:
See Fig. 7.
Collaborative Filtering:
In Fig. 8, the RMSE and MAE values are evaluated by using SVD Algorithm available in Surprise library to minimize the RMSE.
Hybrid Recommender System:
See Fig. 9.
For our hybrid recommender, we get different recommendations for different users although the movie is the same. Hence, our recommendations are more personalised and tailored toward particular users.

```
In [8]: def weighted_rating(x):
            v = x['vote_count']
            R = x['vote_average']
            return (v/(v+m) * R) + (m/(m+v) * C)

In [9]: qualified['wr'] = qualified.apply(weighted_rating, axis=1)

In [10]: qualified = qualified.sort_values('wr', ascending=False).head(250)

In [11]: qualified.head(15)
```

Out[11]:

	title	year	vote_count	vote_average	popularity	genres	wr
15480	Inception	2010	14075	8	29.1081	[Action, Thriller, Science Fiction, Mystery, A...	7.917588
12481	The Dark Knight	2008	12269	8	123.167	[Drama, Action, Crime, Thriller]	7.905871
22879	Interstellar	2014	11187	8	32.2135	[Adventure, Drama, Science Fiction]	7.897107
2843	Fight Club	1999	9678	8	63.8696	[Drama]	7.881753
4863	The Lord of the Rings: The Fellowship of the Ring	2001	8892	8	32.0707	[Adventure, Fantasy, Action]	7.871787
292	Pulp Fiction	1994	8670	8	140.95	[Thriller, Crime]	7.868660
314	The Shawshank Redemption	1994	8358	8	51.6454	[Drama, Crime]	7.864000
7000	The Lord of the Rings: The Return of the King	2003	8226	8	29.3244	[Adventure, Fantasy, Action]	7.861927
351	Forrest Gump	1994	8147	8	48.3072	[Comedy, Drama, Romance]	7.860656
5814	The Lord of the Rings: The Two Towers	2002	7641	8	29.4235	[Adventure, Fantasy, Action]	7.851924
256	Star Wars	1977	6778	8	42.1497	[Adventure, Action, Science Fiction]	7.834205
1225	Back to the Future	1985	6239	8	25.7785	[Adventure, Comedy, Science Fiction, Family]	7.820813
834	The Godfather	1972	6024	8	41.1093	[Drama, Crime]	7.814847
1154	The Empire Strikes Back	1980	5998	8	19.471	[Adventure, Action, Science Fiction]	7.814099
46	Se7en	1995	5915	8	18.4574	[Crime, Mystery, Thriller]	7.811669

Fig. 6 Top 250 movies based on the popularity/user ratings

```
In [32]: get_recommendations('The Godfather').head(10)

Out[32]: 973      The Godfather: Part II
         8387                The Family
         3509                      Made
         4196       Johnny Dangerously
         29              Shanghai Triad
         5667                      Fury
         2412          American Movie
         1582   The Godfather: Part III
         4221                 8 Women
         2159           Summer of Sam
         Name: title, dtype: object

In [33]: get_recommendations('The Dark Knight').head(10)

Out[33]: 7931                    The Dark Knight Rises
         132                           Batman Forever
         1113                          Batman Returns
         8227    Batman: The Dark Knight Returns, Part 2
         7565              Batman: Under the Red Hood
         524                                   Batman
         7901                        Batman: Year One
         2579            Batman: Mask of the Phantasm
         2696                                     JFK
         8165    Batman: The Dark Knight Returns, Part 1
         Name: title, dtype: object

In [34]: credits = pd.read_csv('credits.csv')
         keywords = pd.read_csv('keywords.csv')

In [35]: keywords['id'] = keywords['id'].astype('int')
         credits['id'] = credits['id'].astype('int')
         md['id'] = md['id'].astype('int')

In [36]: md.shape

Out[36]: (45463, 25)
```

Fig. 7 Similar 30 movies based on overview, taglines, director, cast

```
In [65]: svd = SVD()
         evaluate(svd, data, measures=['RMSE', 'MAE'])

         Evaluating RMSE, MAE of algorithm SVD.

         ------------
         Fold 1
         RMSE: 0.8923
         MAE:  0.6891
         ------------
         Fold 2
         RMSE: 0.8982
         MAE:  0.6883
         ------------
         Fold 3
         RMSE: 0.8995
         MAE:  0.6927
         ------------
         Fold 4
         RMSE: 0.9060
         MAE:  0.6967
         ------------
         Fold 5
         RMSE: 0.8935
         MAE:  0.6890
         ------------
         ------------
         Mean RMSE: 0.8979
         Mean MAE : 0.6912
         ------------
         ------------

Out[65]: CaseInsensitiveDefaultDict(list,
                                    {'mae': [0.68907711209711431,
                                      0.68827862853515431,
                                      0.69268970936514584,
                                      0.69674717243046147,
                                      0.68899872532741335],
                                     'rmse': [0.89226612712965325,
                                      0.89823710302444226,
                                      0.89945050602481602,
                                      0.90599131930031884,
                                      0.89346740974468719]})
```

Fig. 8 The RMSE and MAE Values

```
In [76]: hybrid(500, 'Avatar')
```

Out[76]:

	title	vote_count	vote_average	year	id	est
8401	Star Trek Into Darkness	4479.0	7.4	2013	54138	3.561917
8658	X-Men: Days of Future Past	6155.0	7.5	2014	127585	3.365317
974	Aliens	3282.0	7.7	1986	679	3.156818
4347	Piranha Part Two: The Spawning	41.0	3.9	1981	31646	3.058620
2014	Fantastic Planet	140.0	7.6	1973	16306	3.024476
8419	Man of Steel	6462.0	6.5	2013	49521	3.006235
1668	Return from Witch Mountain	38.0	5.6	1978	14822	3.004598
4017	Hawk the Slayer	13.0	4.5	1980	25628	2.999030
922	The Abyss	822.0	7.1	1989	2756	2.970150
1621	Darby O'Gill and the Little People	35.0	6.7	1959	18887	2.923382

Fig. 9 Movie prediction for User ID 500 with movie given as 'Avatar'

5 Conclusion

Upon surveying through different algorithms and models present for movie recommendation systems, we try to come up with a hybrid recommendation algorithm in order to provide the most accurate recommendation to a user based on his ratings and preference.

Recommendation systems of the future will work in e-commerce to offer a more visceral, immersive, and well-rounded experience for every step of a customer's journey. In addition, once a successful hybrid engine comes into the picture, the same algorithm can be extended to other types of recommendation systems. This would

result in an economic boom for the various e-commerce websites and applications with better user-specific experience.

References

1. Kumar, M., Yadav, D. K., Singh, A., & Gupta, V. K. A movie recommender system "MOVREC". https://pdfs.semanticscholar.org/e13e/b41de769f124b3c91771167fb7b01bc85559.pdf.
2. Katarya, R., Verma, O. P. An effective collaborative movie recommender system with cuckoo search https://ac.els-cdn.com/S1110866516300470/1-s2.0-S1110866516300470-main.pdf?_tid=7a06b6fe-c95d-11e7-b816-00000aacb362&acdnat=1510679077_68b65f60dcc55e2aeeacb3588dab49e4.
3. Hande, R., Gutti, A., & Shah, K. Moviemender—A movie recommender system. https://pdfs.semanticscholar.org/e13e/b41de769f124b3c91771167fb7b01bc85559.pdf.
4. Halder, S. Movie recommendation system based on movie swarm. http://www.academia.edu/2396000/Movie_Recommendation_System_Based_on_Movie_Swarm.
5. Xu, Y. Collaborative recommendation with User Generated Content (UGC). https://www.researchgate.net/profile/Yueshen_Xu/publication/280733284_Collaborative_Recommendation_with_User_Generated_Content/links/55c3cf3408aeb97567401ddb/Collaborative-Recommendation-with-User-Generated-Content.pdf.