

# A Movie Recommendation System: Using Content-based and Collaborative Information

<sup>1</sup>Folord Suhas S, <sup>2</sup>Naveen Ganesh, <sup>3</sup>BELLO Mohamed, <sup>4</sup>Laxmi B Rananavare

<sup>4</sup>Professor, <sup>1,2,4</sup>School of C&IT, REVA University <sup>3</sup>Associate Professor, School of C&IT, REVA University <sup>1</sup>folordsuhas@gmail.com, <sup>2</sup>ganeshnaveen9623@gmail.com, <sup>3</sup>bellofad@gmail.com, <sup>4</sup>laxmibranavare@reva.edu.in

Article Info Volume 83 Page Number: 4280-4283 Publication Issue: May - June 2020

#### Abstract

Since the late 20th century, the number of internet users has increased dramatically as has the number of web searches performed on a daily basis and the amount of information available to us. However, not all data that we get in search results are reliable or relevant which means that it may become more and more difficult to get satisfactory results from web searches. To solve this problem we use recommendation systems. Recommendation system shows us only the relevant information. Recommendation Engines can make various suggestions about artifacts to users. In our day-to-day lives, they may predict whether a user may like to buy a particular product online or if they are interested in a particular movie or are interested in listening to a particular song. To pick a movie, a user might search various websites to find a highly rated and well-reviewed movie which is very timeconsuming. Here, we have created this recommendation system using the following methods. Content-based filtering takes keywords from the movie dataset and suggests it to a relevant user. Social based filtering takes reviews from multiple users and suggests it to another user. However, these methods do not use a significant amount of information available. This paper is an approach to a recommendation that is able to use both user ratings and other information available that will help in recommending movies to users. Our method uses these methods on a dataset containing more than 5000 different movies.

# Article History

Article Received: 19 November 2019 Revised: 27 January 2020 Accepted: 24 February 2020 Publication: 12 May 2020

*Index Terms:* Movie recommendation system, Collaborative filtering, content-based filtering, Hybrid approach, Scalability

# 1. Introduction

Entertainment is very important these days, whether it is a busy day at work or just a bad day in general, entertainment such as movies, songs, etc. refresh us and get us ready for the next day. Movies play a huge part in entertaining us. There are many different types of movies such as kids movies, educational movies, comedy, animated movies, etc. Movies can be easily distinguished by their genres. Another way to distinguish movies is by their year of release, language, cast information, etc.,

There are a number of ways in which we can search for a good movie. Recommendation systems have come a long way in this aspect. This system helps us save time and energy hence, they must be reliable and should be able to provide a movie that is exactly what we want or at least similar to what we want. In the last few years, with



the increase in social networking, huge amounts of data are being generated on the internet. To eradicate the overload of data recommendation systems are used as an information filtering tool. Hence, there is a huge potential for exploration in this field.

# 2. Related Work

In 2006, Machine Learning and Data Mining contest was organized by Netflix to improve their recommendation Engines by 10%. Even though their recommendation engines did not improve, the algorithms and techniques discovered were innovative. In 2008 Gaurangi Tilak introduced an expert movie recommendation engine called Movie GEN. They implemented this system using technologies like machine learning and cluster analysis on the basis of hybrid recommendation methods. Based on the user's answers, it refines the movie set and finally recommends movies to the users. Hirdesh Shivhare in 2015, suggested an integrative method by incorporating a fuzzy c-means clustering method and a genetic algorithmbased weighted similarity measure to construct a movie recommendation system.

### 3. Techniques Used in Proposed Methodology

The solution proposed is to improve the quality and simplicity of the recommendation engine. We have used Content-based filtering and collaborative filtering to get the recommendations. To efficiently compute the similarity between the different movies present in the given dataset and to reduce the time taken for computation of the movie recommender system, we have used cosine similarity algorithms.

The overview obtained from the Kaggle dataset is used to detect the similarity between two movies. TfIDvectorizer is implemented to compute the similarity between any two movies in the given dataset from Kaggle.

# A) Weighted hybrid filtering

Weighted technique computes the prediction score as a result of combining all the recommendation approaches by considering them as a variable in the linear combination.

Movies\_Cleaned\_df['weighted\_average'] =((R\*v)+(C\*m)/(v+m))



Figure 1: Graph for Weighted hybrid filtering

# **B)** Content-based filtering

It is also known as cognitive filtering, recommending items based on the comparison between the particular item and the selected user profile. Each item is represented as terms that usually occur in the document. User's profile is also described in the same way as analyzing the contents viewed previously by them.

# C) Collaborative filtering

Collaborative filtering is a technique that can filter out items based on the likes of a similar group of users. Under collaborative filtering, we have,

#### a) Simple SVM (support vector machine)

It is an iterative supervised algorithm. Given a set of training examples, it divides them into one of the two given categories, shown in the figure below.



Figure 2: Support vector machine

### b) Ratings based filtering

This algorithm finds patterns that are consistent across the ratings of the other users. These patterns can be used on their own or in conjunction with social information to recommend content that a user may like.

title	Til There Was You (1937)	1-900 (1994)	101 Daimatians (1996)	12 Angry Men (1957)	187 (1997)	2 Days in the Valley (1996)	20,000 Leagues Under the Sea (1954)	2001: A Space Odynaey (1968)	3 Ninjan: High Noon At Mega Mountain (1998)	39 Steps, The (1935)		Yankee Zulu (1994)	Year of the Horse (1997)	You So Crazy (1994)	Young Frankenstein (1974)	Young Guns (1988)	Young Guns II (1990)	Poi Har Th
user_id																		
0	NaN	NaN	NaN	NaN	NaN	NaN	Nahi	NaN	NaN	NaN	-	NaN	NaN	NaN	NaN	NaN	Nahi	
1	Nati	NaN	2.0	5.0	NaN	NaN	3.0	4.0	NaN	NaN		NaN	NaN	Nati	5.0	3.0	Nati	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	1.0	NaN		NaN	NaN	Nati	NaN	NaN	NaN	
3	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN		NaN	Nahi	NaN	NaN	NaN	NaN	
- 4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN		NaN	NaN	NaN	NaN	NaN	NaN	
5 rows ×	1664 o	olumns																

Figure 3: Pivot table used by ratings based filtering

# D) Cosine similarity measure

The "cosine similarity" measure for two vectors is calculated by taking a measure of the cosine of the angle between them. This checks for the orientation of the vectors, not their magnitude.

Formula for "cosine similarity":

$\vec{a}\cdot\vec{b}=\ \vec{a}\ \ \vec{b}\ \cos\theta$	
$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\ \vec{a}\  \ \vec{b}\ }$	

It is a criterion used to estimate how much identical objects (here movies) are to each other.

*Euclidean's distance* can measure how dissimilar users are to each other when they don't like similar movies, but cannot be implemented when users like similar movies. *Cosine Similarity* overcomes Euclidean's distance problem of measuring similar users when 2 users like the same movies and dislike the same movies.



### E) Tf-IDF Vectorizer

In content-based filtering, TF-IDF is used to provide priority (importance) to the words present in the summary. The priority is inversely related to the occurrence of the words in the passage. ie. The uncommon words have more importance than the common words.

The basic formula is

TF - IDF = TF \* IDF

Where TF is Term frequency

IDF is Inverse document frequency

*Tf-IDF* Vectorizer helps overcome the bag of words methodology which assigns almost the same priority levels to every word irrespective of how many times they occur in a document(overview column).

#### F) Pearson's Correlation

Pearson's Correlation is used to find the strength of the covariance between 2 quantities (ie. Movies and User ratings). It is implemented for recommendations in collaborative filtering.

The value of the coefficient always lies between the range 1 and -1, ie 1 being the most similar movies and -1 being least similar movies (in this case).

Pearson Correlation Formula :

 $(x, y) = Covariance (x, y) / (\sigma x * \sigma y)$ 

Where x and y here are movie names, and user ratings.

#### 4. Proposed Methodology

#### A. Dataset information

One dataset from Kaggle and two other datasets from MovieLens have been used which have been generated for the purpose of research in the recommendation field. These are:

1) Kaggle 5k movies

2) Kaggle 5k credits

3) MovieLens 5k latest

The datasets have the following features:

1. Ratings are assigned from 1 to 5. (1 means very bad, 5 means very good).

2. A minimum of 15 movies have been rated by each user.

3. Other simple information such as gender, age, address, etc. is also provided.

4. Movie features such as cast info, production info, etc. are also included.

The reason for including datasets from multiple sources is to test the scalability of the engine, that is even with different datasets, the system will continue working well and give a good performance.

#### 5. Screenshots & Expected Outputs

Providing Input for Content-based Filtering: name of the movie

```
In [21]: def give_rec(tite, sig-sig):
    # Get the index corresponding to original_title
    idx = indics[titia]
    # Get the poirwaits similarity scores
    sig_scores = list(enumerate(sig[idx]))
    # Sort the movies
    sig_scores = sorted(sig_scores, key-lambda x: x[1], reverse=True)
    # Sort the movies
    sig_scores = sig_scores[1:1]
    # Movie indices
    movie_indices = [i[0] for i in sig_scores]
    # Toty is movies
    return movies_cleaned_off[original_title'].iloc[movie_indices]
In [22]:    # Testing our content-based recommendation system with the seminal film Spy Kids
    give_rec("Pirates of the Caribbaan: At World's End")
```

Name of the Movie input by user

Providing Input for Collaborative Filtering: (name, rating) of the movie



Output: Movies Recommended (ranked top 20 Most Similar Movies)

Out[10]:	Mission: Impossible III (2006)	1.500000
	I, Robot (2004)	0.854793
	Chronicles of Riddick, The (2004)	0.813305
	Snakes on a Plane (2006)	0.809323
	Mission: Impossible - Ghost Protocol (2011)	0.799121
	King Arthur (2004)	0.779185
	Flightplan (2005)	0.765644
	Last Samurai, The (2003)	0.761290
	Machete (2010)	0.760784
	Paycheck (2003)	0.753726
	Mission: Impossible II (2000)	0.752457
	Day After Tomorrow, The (2004)	0.746326
	Island, The (2005)	0.744711
	War of the Worlds (2005)	0.743728
	Wedding Crashers (2005)	0.736321
	King Kong (2005)	0.735294
	Terminator 3: Rise of the Machines (2003)	0.734056
	Over the Hedge (2006)	0.728020
	X-Men: The Last Stand (2006)	0.726114
	Crank (2006)	0.726011
	dtype: float64	

#### 6. Conclusion

In this research paper, we have improved the accuracy, stability, and performance of the recommendation engine. We used collaborative filtering and content-based filtering to try and achieve the best performance. We compare multiple datasets to find out if the system is consistent with different datasets.

#### 7. Future Work

The research work was successfully completed and we're satisfied with the results. Based on that, the following features can be added in the future:

1) Language filters can be applied to further enhance the recommendations.

2) Different categories of movies can be generated.

3) Users can find out if they've already watched the movie.

#### Acknowledgment

The success and final outcome of this project required a lot of guidance and assistance and we are extremely



privileged to have got all this support. We wish to acknowledge the help provided by our professors from the Computer science department of REVA University.

# References

[1]	Base paper							
	reference:https://ieeexplore.ieee.org/document/8							
	058367/keywords#keywords							

- [2] Dataset reference:
- https://grouplens.org/datasets/movielens/
- [3] Google News Personalization: Scalable Online Collaborative Filtering.
- [4] https://www.kaggle.com/tmdb/tmdb-moviemetadata
- [5] iopscience.iop.org/#
- [6] SajalHalder, A. M. Jehad Sarkar, Young-koo Lee "Movie recommendation system based on movie swarm" published in 2013 IEEE conference.
- [7] Ching-Seh Mike Wu, Deepti Garg, Unnathi Bhandary "Movie recommendation system using collaborative filtering" published in 2018.
- [8] "A Hybrid Approach using Collaborative filtering and Content-based Filtering for Recommender System" G Geetha, M Safa, C Fancy et al.
- [9] S.V.N. Vishwanathan, M. Narasimha Murty [2002], "SSVM: A Simple SVM Algorithm, Neural Networks, IJCNN '02. Proceedings of the International Joint Conference on Volume 3"
- [10] Eyrun A. Eyjolfsdottir, Nan Li and GaurangiTilak [2008], "MovieGEN: A Movie Recommendation System"
- [11] ShiladSen, J. Ben Schafer, Jon Herlockerand DanFrankowski [2007] "Collaborative filtering recommender systems"
- [12] F. Kong, X. Sun, S. ye, [2005], "A comparison of several algorithms for collaborative filtering in startup stage"
- [13] J.L. Sanchez, F. Serradilla, E. Martinez, J. Bobadilla, [2008] "Choice of metrics used in collaborative filtering and their impact on recommender systems"
- [14] "Deep Matrix Factorization for Recommender Systems with Missing Data not at Random" Fei Zhang, Shiyu Peng and Jiaxing Song.
- [15] Joseph Konstan, George Karypis, Badrul Sarwar, and John Riedl, (2001) "Item-Based Collaborative Filtering Recommendation Algorithms".