Movie Recommendation System

S. USHA PRIYA
Computer Science Department
PES University
Bangalore, India
ushapriya525@gmail.com

Shamitha S
Computer Science Department
PES University
Bangalore, India
sshamitha770@gmail.com

Guruprasad N Bamane
Computer Science Department
PES University
Bangalore, India
guruprasadn.1402@gmail.com

INTRODUCTION AND BACKGROUND

Ever in this busy world amidst the work throttle there has always been a need for the source of entertainment. This might be habitual or rarely occurring which might bring in a state of pleasant mindedness. There are many forms of entertainment and methods followed upon like films, music, dance, and video games, etc... The springs arising depend on a particular individual and vary from time to time. One among these could also be a habit of a person based upon their interest. Even though there are many existing forms of killing time, the one which definitely has its place is movies or films.

With an ever-increasing diversity and technology, one can see many kinds of motion pictures being available. It could be something like a short film, an illusionary , a prediction made, or something purely based for entertainment. The classification may also go in accordance with the timeline the movie was released . The grouping can also be based on genre like Western, Horror, Action, Drama and so on. People might prefer flicks based on the time which they can spend upon it, those with a heck of a schedule may tend towards one's which have shorter duration. Some particular genre or a language can be attractive which again depends on perspective.

Existing in this physical world, it is troublesome and time consuming to surf among people or sites to decide upon which show to spend time on. One would rather like someone to suggest things based on their interest. But this wouldn't be a realistic approach due to the huge number of options and scarcity of time.

Solution which doesn't require a constant manpower to rely upon would be suitable. This is exactly what most of the applications and websites like Netflix, Amazon prime and hot-star do. This would always give raise to an eagerness of how exactly these technologies come up with the right recommendations.

It is important to understand how the recommendations are made and the underlying principles behind it. Recommendations basically fall under the category of unsupervised learning in machine learning, wherein a model is trained upon a dataset which itself categorizes the available data into categories based on similar features . This model later can be made to predict further views.

Though there are a wide range of models and works done under this field , our main area of interest is to make comparisons among possible methods and to come out with a conclusion on which method would mostly comply with the given situation. Our research mainly focuses on making recommendations using different methods available, drawing insights between

variables and comparing the methodologies used.

Keywords: Recommendation, Content-based, Collaborative filtering, KNN, Cosine Similarity, Pearson's coefficient

RELATED WORK

Basically, Movie recommendation system might suffer from problems like long-tail problem, Sparsity, and cold-start problem[3]. The technique used was Collaborative filtering(CF) with Pearson's correlation coefficient. Experiment was conducted on MovieLens-100k dataset. All the movies which had ratings less than the average i.e., average between 0 and maximum, are ignored. If the provided ratings by users were wrong, then it can neglect a good movie.

CF comprises user-based as well as item-based filtering [1]. These two approaches are used to make recommendations on the Movie Lens dataset , taking rating as the only parameter into consideration for making recommendations based on the similarity among various users. Performance metrics like Pearson's correlation coefficient, cosine similarity, Euclidean distance and Jaccard similarity are used. According to the experimental results item–based(0.84) CF led to greater accuracy compared to user-based filtering(0.76). Also, for diverse datasets Pearson's correlation is one of the best similarity measures. Instead of using only CF, content based should also be used to make comparisons.

Recommendations are made based on the history and choices that a user makes. [2] Content- based recommendation(CBR) is used for sentiment analysis and CF is used to give recommendations. The model works on user ratings finding the similar ones which the user might like. K nearest neighbour(KNN) and cosine similarity are used as metrics for evaluation. The recommendation framework uses the user's minimum rating to assess the type of film to recommend. This is a function that has not been used in prior programs of a similar nature. KNN might not work for large datasets.

Due to the day by day increase in the amount of information as well as the census there is a compromise seen in movie recommendation systems. [4] Hybrid approach consisting of both content-based filtering (CBF) and CF would rather help improve accuracy, quality, computing time and scalability. Support vector classifiers and Genetic algorithms are being used. Three different datasets are used as a combination
Accuracy, precision, recall and F1 score are comparatively higher for a hybrid approach. There is a need to work on the memory requirements of the proposed approach.

A machine learning approach to recommend movies to users using K-means clustering algorithm to separate similar users and creating a neural network for each cluster[5]. Machine learning approaches are used to guess what rating a particular user might give to a particular movie so that this information can be used to recommend movies to viewers. User rating, user consumption ratio and user preference have been considered while building the system. K-means clustering has been used to separate users with similar taste in movies. This approach is not suitable in handling noisy data and data containing outliers. Also, the number of clusters 'k' values needs to be mentioned beforehand.

In this global world there is an adamant increase in the amount of data getting generated and users on the other hand have no energy and time to spend on searching. Though there are many movie recommender systems there is also a need to accommodate large data in this domain.

Distributed cloud computing using map reduce framework would suffice greatly[7]. Recommendations are made based on the past views of a user i.e., item-based CF technique is used. Assumption as in the same framework can be used also in ecommerce.

Assumptions made with respect to our problem statement:
Some columns like budget, runtime, revenue, homepage, original language, original title, production companies, production countries , release date, spoken languages, status, tagline are dropped as they do not contribute towards CBR or CF to a greater extent.
In CBR instead of taking genre, crew, caste, and keywords segregated, they are combined and made together as tags attributes as we assumed that aggregating them would lead to a better recommendation.
In CF to account for the popularity of the movie a threshold value is set to 100, i.e., whichever movie is being rated by more than 100 people is considered as a popular one.
In CBF using weighted average scores only movies greater than 70 percentile votes are considered for making recommendations as they are popular and are useful.

Recommendations by models are done only for movies which are already existing in the dataset as it is the way how recommendations are proposed.

PROPOSED SOLUTION AND IMPLEMENTATION

This section describes the steps of implementation of fake news detection project through models, and how it is connected to the interface
Overall, there are 5 models built in this project.

CBF using weighted average scores

Pre-processing:

Two datasets, movies and credits are read from official tmdb datasets and are checked for null values in each of the columns. There are 3091 null values in the homepage and 844 nulls in the tagline. The missing values in the home page and tagline will not affect our analysis as we are not focusing on these columns.
Credits and movies datasets are merged together based on the title column. Columns like budget, runtime, revenue, homepage, original language, original title, production companies, production countries , release date, spoken languages, status, tagline are dropped as they do not contribute towards CBR or CF to a greater extent. Duplicates are dropped based on the values in all columns.

Transformation:

Genres and keywords column in movies dataset is converted into a list of comma separated values using an user defined conversion function for further computation.

Model:

All movies with vote count greater than 70 percentiles are taken, and weighted average is computed as, W = (Rv+Cm)/(v+m)

W=Weighted Rating, R=rating, v=average votes, m=minimum votes, C=mean vote across the whole product.

Sorted ranking is taken based on weighted average in descending order, and popularity is also sorted in descending order.

Normalization:

Using MinMaxScalar both weighted average and popularity are normalized.

This is a way to analyse the most popular movies in the entire dataset taking all factors into consideration.

CBF based on user interest:

Pre-processing:

Movies and credits datasets are checked for null values. Both datasets are merged based on title. Irrelevant columns (budget, runtime, revenue, homepage, original language, original title, production companies, production countries , release date, spoken languages, status, tagline) are dropped. All duplicates are dropped.

Transformation:

Overview column is transformed into a list of commas separated values.

Model:

Top 3 actors are retrieved from the cast and the value of the director role is taken from crew to be considered. Overview, genres, keywords, cast and crew are combined into tags for recommendation. The Tags column is stemmed using PorterStemmer and vectorized using CountVectorizer.

Pairwise similarity is computed using cosine similarity among all the movies based on tags.

Evaluation:

Five movies to an already existing movie in the dataset are recommended based on descending order of their similarity.

Content based Filtering using dot product

Pre-processing:

Two datasets, movies and ratings are considered. Titles and genres in movies are formatted using split and strip. Separate data-frame of movies is created, and each genre is added as a column, a value 1 is appended as a value in the cell if a particular movie belongs to a specific genre. Nan values are filled with 0 to show that a movie doesn't have a particular genre. Year is dropped from the dataset as it is irrelevant.

Model:

The new created data-frame undergoes a dot product with ratings data-frame to get weights. Genres are multiplied by the weights and then weighted average is taken, this is stored as recommendation. Recommendation is sorted in descending order.

Evaluation:

Each movie id is taken as a parameter and 20 other movies are recommended for each based on recommendation computed above.

Collaborative Filtering(User-user filtering. Recommendation based on similar other users)

Pre-processing:

Year column is dropped as it doesn't carry much weight in recommendation. Title column is formatted. Genre attribute is also removed as in CF we aren't making recommendations based on a particular genre, rather we focus on ratings.

Movies and ratings data-frames are merged based on the title.

Model:

KNN and cosine similarity are used for CF. Total ratings are counted for each and every movie. This column is merged into the final data-frame for consistency.

Popularity threshold is set to 100 and a separate data-frame with all movies having rating count greater than 100 are taken into consideration.

Transformation:

A pivoted table is created from the above data-frame and is converted into an array for similarity computation.

KNN based on cosine similarity is used in identifying the six nearest neighbours and thereby recommended.

Collaborative Filtering with Pearson Coefficient(using SciPy)

Pre-processing:

Genres and timestamps are dropped as they aren't needed. Dataset is pivoted with user id as rows and title as column.

Model:

A correlation matrix is computed based on Pearson's correlation coefficient between all movie titles.

Evaluation:

For a particular movie 20 similar movies are returned in descending order of their correlation.

EXPERIMENTAL RESULTS

This section contains all outcomes the research has predicted under two different circumstances of hardware and varying inputs on software.

- *Dataset Description:*

Two datasets tmdb[7] and movies[8] are considered with each one having another 2 datasets. Tmdb has tmdb_5000_movies and tmdb_5000_credits each having (4804,20) and (4814,4) dimensions respectively. Movie's dataset has movies and ratings with (9743,3) and (100827,4) rows and columns respectively.

Vote_count and vote_average are dependent variables; all the other attributes are independent.

Two main datasets with their respective attributes are shown below in table 1 and table 2

Table 1:Attributes of tmdb dataset with its respective data-types

COLUMN	DATATYPE
Index	Integer
Budget	Integer
Genres	Object(List)
Keywords	Object(List)
Overview	Object(String)
Popularity	Float
Title	Object(String)
Vote_average	Float
Vote_count	Integer
Cast	Object(List)
Crew	Object(List)

Table 2:Attributes of movies dataset with its respective datatypes

COLUMN	DATATYPE
Index	Integer
MovieId	Integer
userId	Integer
Rating	Float
Title	Object(String)
Genre	Object(List)

- *System Specifications:*
  All models are tested on systems of two different specifications as shown below in Table 3:

Table 3: System Specifications

Manufactured company	Asus 8th Gen	Dell G3 10th Gen
Processor specification	Intel i5	Intel i7
RAM Capacity	8GB	16GB
Operating System	Windows 10 64-bit	Windows 10 64-bit
IDE	Jupyter Notebook & Spyder	Jupyter Notebook & Spyder

	Anaconda 3	Anaconda 3

On both machines of the above specification respectively, the model yielded exact results and successfully ran through all the edge case conditions.

Comparison of all genres with respect to the metrics like frequency, vote_average, vote_count, average popularity, average_budget, average_revenue is shown in Figure 1 below.

	Geners	Frequency	Average_Vote_Average	Average_Vote_Count	Average_Popularity	Average_budget	Average_revenue
0	Action	1154	5.989515	1135.814558	30.940382	51.510751	141.213098
1	Adventure	790	6.156962	1430.669620	39.268042	66.326861	208.660204
2	Fantasy	424	6.096698	1323.983491	36.387043	63.560605	193.354245
3	Science Fiction	535	6.005607	1425.585047	36.451806	51.865551	152.455515
4	Crime	696	6.274138	752.599138	22.853274	27.849808	66.150662
5	Drama	2297	6.388594	532.280801	17.764853	20.678325	52.116232
6	Thriller	1274	6.010989	772.845369	24.460077	31.966207	81.044291
7	Animation	234	6.341453	1246.170940	38.813439	66.465902	225.693025
8	Family	513	6.029630	902.260448	27.832849	50.719512	162.345495
9	Western	82	6.178049	607.853659	18.236279	27.078702	46.245965
10	Comedy	1722	5.945587	525.181185	18.221001	25.313421	71.289499
11	Romance	894	6.207718	454.504474	15.962426	20.311362	60.002391
12	Horror	519	5.626590	495.211946	18.295915	14.574031	43.545076
13	Mystery	348	6.183908	799.146552	24.586827	30.744487	78.300927
14	History	197	6.719797	486.791878	17.444839	29.903467	57.523562
15	War	144	6.713889	756.645833	23.777289	35.282457	84.155874
16	Music	185	6.356676	300.497297	13.101512	15.907948	48.455952
17	Documentary	110	6.238182	70.590909	3.945724	2.653288	9.838888
18	Foreign	34	6.352941	10.764706	0.686787	0.658088	0.364652
19	TV Movie	8	5.662500	209.750000	6.389415	1.150000	0.000000

Figure 1:Heatmap between genres and various metrics

The above heat-map implies that 'Adventure' and 'Animation' dominate in vote_count, popularity, budget and revenue whereas 'Drama' counts for highest frequency.

The below correlation plots in Figure 2 show that there is a positive correlation between average_popularity and average_vote_counts and there is weak positive correlation between average_revenue and average_budget.

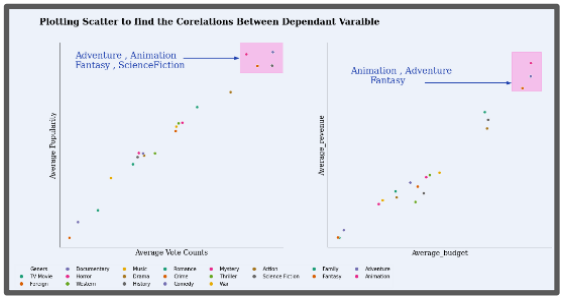

Figure 2: Correlation plots between dependent variables

CBF using weighted_scores

Figure 3 below shows weighted average scores based on vote_count and vote_average of top 20 scores.

	original_title	vote_count	vote_average	weighted_avg	popularity
1887	The Shawshank Redemption	8205	8.5	8.340847	136.747729
3342	The Godfather	5893	8.4	8.192985	143.659698
662	Fight Club	9413	8.3	8.171711	146.757391
3237	Pulp Fiction	8428	8.3	8.157685	121.463076
65	The Dark Knight	12002	8.2	8.102725	187.322927
809	Forrest Gump	7927	8.2	8.056133	138.133331
1824	Schindler's List	4329	8.3	8.038876	104.469351
3872	Whiplash	4254	8.3	8.034826	192.528841
96	Inception	13752	8.1	8.018655	167.583710
1996	The Empire Strikes Back	5879	8.2	8.010524	78.517830
2300	千と千尋の神隠し	3840	8.3	8.009994	118.968562
95	Interstellar	10867	8.1	7.998155	724.247784
2737	The Godfather: Part II	3338	8.3	7.972846	105.792936
329	The Lord of the Rings: The Return of the King	8064	8.1	7.965134	123.630332
2917	Star Wars	6624	8.1	7.938179	126.393695
690	The Green Mile	4048	8.2	7.935576	103.698022
1559	Se7en	5765	8.1	7.916275	79.579532
262	The Lord of the Rings: The Fellowship of the Ring	8705	8.0	7.880700	138.049577
1853	GoodFellas	3128	8.2	7.869987	63.654244
2097	The Silence of the Lambs	4443	8.1	7.867931	18.174804

Figure 3:Best movies by votes

Graph in figure 4 below indicates the best movies by weighted scores.

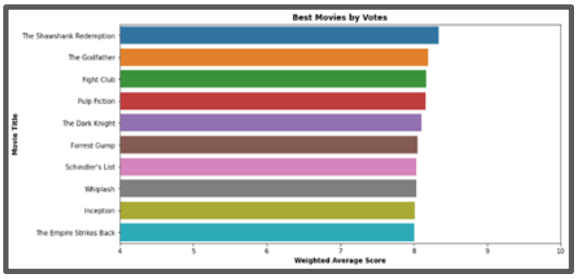

Figure 4: Best movies by weighted_score

Graph in figure 5 below indicates the best movies by popularity.

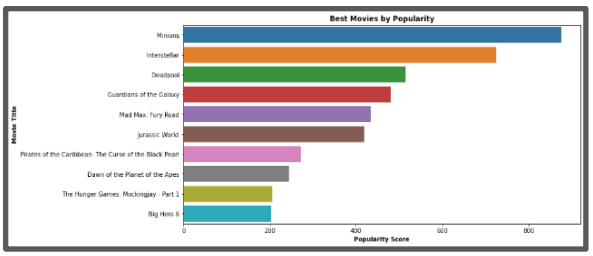

Figure 5:Best movies by popularity

The given visual in figure 6 portrays the best movies by both weighted scores and popularity.

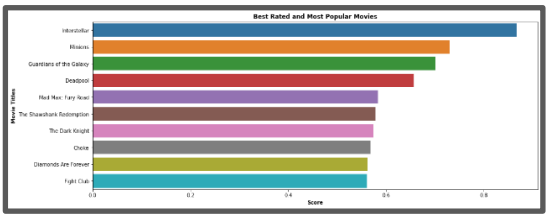

Figure 6:Best movies by weighted_scores and popularity

CBF based on user-interest

```
similarity[5][4]
#similarity of movie at 5th index to 4th movie
0.08186216918157273

similarity[5][6]
#similarity of movie at 5th index to 6th movie
0.10219250343291215

similarity[5][10]
#similarity of movie at 5th index to 10th movie
0.19174124721184263

similarity[5][12]
#similarity of movie at 5th index to 12th movie
0.21741413836966814
```

Figure 7: Similarity values with respect to movie 6

The above picture in Figure 7 represents similarity between movie 6 and other movies 4,7,11 and 13 respectively.

```
similarity[5][5]
#similarity of movie at 5th index to 1st movie
0.9999999999999993
```

Figure 8:Similarity between movie 6 with itself

As shown above in Figure 8 every movie is like itself.

Recommender System using K-Nearest Neighbours and cosine similarity

```
# Taking a new movie at random
query_index = np.random.choice(features.shape[0])    # Collect 1 record
print(query_index)

57

# Find similar movies(nearer to the selected movie) using kneighbors
distances, indices = model.kneighbors(features.iloc[query_index,:].values.reshape(1, -1), n_neighbors=6)
# n_neighbors = 6 => will include the movie itself => We will be getting 5 other movie recommendations

# distances = 0 => Same movie itself
distances

array([[8.881784200e-16, 1.782273590e-01, 3.351590580e-01, 3.794637500e-01,
        4.046831970e-01, 4.113857090e-01]])

indices

array([[ 57,  58,  61,  94, 117,  44]], dtype=int64)
```

Figure 9: K nearest neighbours with movie index 57

The above picture in Figure 9 shows 6(k) nearest neighbours to the movie at index 57. It also shows the distances between them.

```
#Top most user with id 91 having all 5 similar movies watched
userSubsetGroup[0]

(91,
       movieId                title  year  userId  rating  TotalRatingCount
 32          1            Toy Story  1995      91     4.0               215
 226         2              Jumanji  1995      91     3.0               110
 7905      296          Pulp Fiction  1994      91     4.5               307
 28816    1274                Akira  1988      91     5.0                39
 37388    1968  Breakfast Club, The  1985      91     3.0               113)
```

Figure10:top users like id 91

This image in Figure 10 above shows top users with id 91 having all 5 similar movies.

CBF with Pearson Coefficient(using SciPy)

```
action_lover = [("Amazing Spider-Man, The (2012)",5),("Mission: Impossible III (2006)",4),
                ("Toy Story 3 (2010)",2),("2 Fast 2 Furious (Fast and the Furious 2, The) (2003)",4)]
similar_movies = pd.DataFrame()
for movie,rating in action_lover:
    similar_movies = similar_movies.append(get_similar(movie,rating),ignore_index = True)

similar_movies.head(10)
similar_movies.sum().sort_values(ascending=False).head(10)
#example

Amazing Spider-Man, The (2012)                  3.233134
Mission: Impossible III (2006)                  2.874798
2 Fast 2 Furious (Fast and the Furious 2, The) (2003)    2.701477
Over the Hedge (2006)                           2.229721
Crank (2006)                                    2.176259
Mission: Impossible - Ghost Protocol (2011)     2.159666
Hancock (2008)                                  2.156098
The Amazing Spider-Man 2 (2014)                 2.153677
Hellboy (2004)                                  2.137518
Snakes on a Plane (2006)                        2.137396
dtype: float64
```

Figure 11:pearson Coefficient

Above picture in Figure 11 displays top 10 movies which are like the movies in the user's history, based on the ratings given by other users.

On what cases does the model work well

All CBR and CF models designed as a part of this project work well on recommending movies to a user if the

user exists in the dataset and has already watched some movie in the dataset. Also, If the model is based on CF, it works when the movie watched by a target user is also watched by some other users.

Where does it fail?

Cold start problem exists in both CF and CBR i.e., if a new movie has been added and is not rated by any user then recommendations wouldn't suffice. Cold start problem also occurs if a new user is added who doesn't have a past history.
Each time a new movie or a user is added, similarity metrics need to be recomputed.

CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have proposed various solutions to the real time problem of movie recommendation. Our specified solution successfully makes appropriate recommendations given a movie based on different similarity metrics.
CBR models don't need any data about other users since the recommendations are specific to one user. This model can capture the specific interests of a user, and can even recommend nice movies that a very few users are interested in.
When it comes to CF domain knowledge is not required because the embeddings are automatically learned. The model can help users discover new interests.
Future work might be focusing on building ensemble models which would give even more suitable recommendations. Knowledge based recommendation systems can be used, taking some of the user requirements into consideration.

LITERATURE STUDY REFERENCES

[1] Analysis of Movie Recommendation Systems with and without considering the low rated movies. Muppana Mahesh Reddy, R. Sujithra Kanmani, Dr. B. Surendiran -2020
https://ieeexplore.ieee.org/abstract/document/9077803

[2] Similarity Based Collaborative Filtering Model for Movie Recommendation Systems. Raghavendra, C K Srikantaiah K.C -2021

https://ieeexplore.ieee.org/abstract/document/9432354

[3] User Centric and Collaborative Movie Recommendation System Under Customized Platforms. Souptik Saha, S.Ramamoorthy, Eisha Raghav – 2021

https://ieeexplore.ieee.org/abstract/document/9451672

[4] An Improved Approach for Movie Recommendation System **.** Shreya Agrawal, Pooja Jain – 2017
https://ieeexplore.ieee.org/abstract/document/8058367

[5] Movie Recommendation System Using Clustering and Pattern Recognition Network. Muyeed Ahmed, Mir Tahsin Imtiaz, Raiyan Khan – 2018
https://ieeexplore.ieee.org/abstract/document/8301695

[6] Movie Recommendation System Employing the User-based CF in Cloud Computing. Tianqi Zhou, Lina Chen, Jian Shen-2017
https://ieeexplore.ieee.org/abstract/document/8005971

Datasets:
[7] https://www.kaggle.com/tmdb/tmdb-movie-metadata
[8] https://drive.google.com/file/d/1WWQCl9w52M1sXNWd4JSKL7q-HHywk03p/view

CONTRIBUTIONS:

Guruprasad N Bamane	PES1UG19CS 174	Building collaborative filtering models, Literature study, Final report
S Usha Priya	PES1UG19CS 407	Pre-processing, Visualizations, Literature study, Final report.
Shamitha S	PES1UG19CS 448	Building Content based filtering models, Visualizations, Literature study, Final report.

REFERENCES:

- https://builtin.com/data-science/collaborative-filtering-recommender-system
- https://realpython.com/build-recommendation-engine-collaborative-filtering/
- https://www.analyticsvidhya.com/blog/2015/08/beginners-guide-learn-content-based-recommender-systems/
- https://en.wikipedia.org/wiki/Recommender_system
- https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0
- https://matplotlib.org/stable/gallery/misc/load_converter.html