Movie recommendation System using TF-IDF based Cosine Similarity approach and Gaussian Mixture Model with Expectation Maximization approach

Sanket J Shah, Kishan Mehta, Harsh Patel, Dhruv Sharma

***Abstract*—The amount of media content is increasing exponentially everyday. All the platforms that provide the content have a 'recommended' section where they provide relevant media content that the user might like. In this report we devise a method of finding these relevant 'recommendations' with the help of machine learning. The machine learning approach used here is the similarity approach, where the recommendations are based on the similarity of the features between the movies. The similarity metric used is the 'cosine similarity'. The literature review, implementation and the results for the same are discussed in this report.**

***Index Terms*—Movie recommendation system, machine learning, cosine similarity, tf-idf, content recommendation.**

I. Introduction

WITH the increasing amount of content being made available on internet everyday, any content consumer would naturally feel overwhelmed by it. According to 2020 Global Internet Phenomena Report, Major OTT platforms like Netflix, Hulu, Amazon Prime and others are responsible for 65% of the total Internet traffic [6]. This content overflow has both pros and cons. Even though it allows users to access any content from around the world at anytime, it makes the search for content of user's liking difficult. Hence, there needs to be a system which helps people guide through the endless stream of visual content to find their choice of movie or TV show.

Even though it seems like an easy task, there is more than what meets the eye. There are many factors which need to be considered while developing such system. An user might prefer a movie or TV show because of its cinematography, actors, genre or even music. An ideal recommendation system needs to recognize the pattern in user's history and learn from it. It needs to understand what the user prefers and suggest future content based on it. Therefore, the most suitable solution would be to use a machine learning model that can learn from user's past history. Hence, this report talks about the various machine learning models that can be used for the same.

II. Literature Survey

A. Cosine similarity

Cosine similarity is also known as the cosine co-efficient. It is a similarity metric that uses the angle between two vectors to find the similarity between them. Consider vectors v1 and v2 that have an angle of θ between them as shown in the figure below. The cosine similarity between them would be $cos(\theta)$. [7]

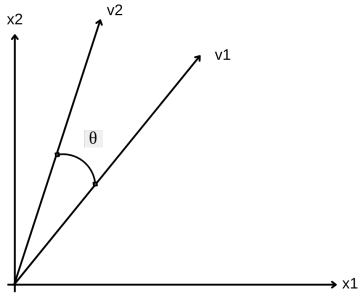

Fig. 1: Cosine similarity

For two matrices, it is defined as follows:

$$cosine(x, y) = \frac{x.y^\intercal}{||x||.||y||}$$

B. TF-IDF Vectorizer

TF-IDF also known as Term Frequency-Inverse Document Frequency is a method used to retrieve information from a given text. It consists of two terms.
1. Term-Frequency is a ratio of given word's frequency out of total distinct words.
2. Inverse Document Frequency is a logarithmic ratio of total number of texts and texts with given term in it.
The TF-IDF value is generated by multiplying both of these terms. [5]

$$TF_{i,j} = \frac{n_{i,j}}{\Sigma_k n_{i,j}}$$

(a) TF

$$IDF\ (wt) = log(\frac{N}{df_i})$$

(b) IDF

$$TF - IDF(wt) = TF_{i,j} \times IDF(wt)$$

(c) TF-IDF

Fig. 2: TF-IDF Formula

C. Gaussian Mixture Models

Gaussian Mixture Model [4]. is a very well known technique used in soft clustering. GMM is used in recommendation systems to precisely discover a user's interests and degree of preference on an item to better recommend items to the user in future.

Each GMM consists of k Gaussian distributions. Among them, each distribution are linearly added together to form the probability density function of GMM

For a random vector x in n-dimensional sample space, if x obeys Gaussian distribution, the probability density function is as follows:

$$p_M(x) = \sum_{i=1}^{k} \alpha_i \cdot p(x \mid \mu_i, \Sigma_i)$$

Fig. 3: Probability Density Function

where μ is an n-dimensional mean vector and Σ is a covariance matrix of nxn(kxnxn in our case) and α is the corresponding mixing coefficient.

D. Expectation Maximization

Expectation Maximization [1] is an algorithm used to map clusters on the unsupervised data points in GMM. The algorithm is divided into two steps:
Initially, any random mean, variance and weights are assigned to the gaussian distributions.
E-step: Assign a soft probability to each of the data points(Ric) based on the distributions. This soft probability values is also known as responsibilities suggests how close any given data point is to the clusters.
M-step: The mean, variance and weights of all the distributions are recalculated based on the responsibility matrix obtained in the E-step.
This algorithm is then called iteratively for specified number of times and finally the Gaussian Mixture Model for the given data points is obtained.

$$r_{ic} = \frac{\text{Probability Xi belongs to c}}{\text{Sum of probability Xi belongs to } c_1, c_2, .. c_k} = \frac{\pi_c \mathcal{N}(x_i \; ; \; \mu_c, \Sigma_c)}{\sum_{c'} \pi_{c'} \mathcal{N}(x_i \; ; \; \mu_{c'}, \Sigma_{c'})}$$

Fig. 4: Responsibility Function

III. Implementation

A. Cosine similarity

For implementation, the dataset that we used is the 'MovieLens 25M Dataset' [3]. The approach to implementation was to use the different features of the movies and find the similarity between them. The following combination of the features were used to get the results:
1. movieTagline + movieOverview
2. movieGenre + movieDirector
3. Movie MetaData (movieDirector + movieCast(Top 3 names from the data) + movieGenre + movieKeywords)

The 'term frequency–inverse document frequency(TF-IDF) vectorizer' is used to convert the string input to numeric input so that the calculation becomes efficient. The TF-IDF vectorizer is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

The following are the steps for implementation of 1. movieTagline + movieOverview:
1. Fetch the movieTagline and the movieOverview columns from the given dataset.
2. Combine the above columns to make a new 'movieDescription' column.
3. Use the TF-IDF vectorizer to convert string to numeric data.
4. Use the cosine similarity to get the pairwise similarity matrix for all the movies.
5. Extract the top n movies with the highest similarity index with respect to the input movie.
6. Evaluate the output qualitatively.

B. Gaussian Mixture Model with Expectation Maximization

The "MovieLens 100K Dataset" [2] for this approach. For this approach "U.data" and "U.user" data frames were used, these data frames contain the user id,information about the users, movie id and the ratings given by the users to those movies. First this data was cleaned and a matrix was created, where each row depicts the rating given by a user to different movies. The entries which were missing "NaN" replaced with "0". This matrix was then passed to the GMM model. We tried two approaches for the Gaussian Mixture Model, for the first we used the model provided by the SciKit library of python and for the second we tried to implement the model of GMM on our own.

Following are the steps for implementing GMM with EM.
1. Choose the desired number of distributions and initialize them with any arbitrary mean, variances and equal weights.
2. E step: Calculate the responsibilty matrix for all the given data-points.
3. M-step: Now, recalculate the mean, variances and the weights for all the gaussians using the responsibility matrix.
4. Repeat the above E and M steps for the specified number of iterations.
5. Finally, we now have the means, weights and a covariance matrix for the fitted Gaussian Mixture Model.
6. Then we calculated the responsibility matrix again to find the cluster that each data point(user's rating) fits best to.
7. Using this information we assign the mean of that cluster as the new rating given by that user.
8. We then use the original Rating matrix to replace all the non-zero ratings to this new Predicted matrix.

IV. Results

A. Cosine similarity

The results obtained are evaluated qualitatively.

1) movieTagline + movieOverview

```
1 get_recommendations('The Dark Knight').head(20)

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
6144                               Batman Begins
7933          Sherlock Holmes: A Game of Shadows
5511                              To End All Wars
4489                                       Q & A
7344                        Law Abiding Citizen
7242                  The File on Thelma Jordon
3537                              Criminal Law
2893                              Flying Tigers
1135                       Night Falls on Manhattan
8680                           The Young Savages
```

Fig. 5: Top 20 similar movies

The above figure shows the results for the input movie 'The Dark Knight'(A batman movie). The results recommend all the other batman movies at the top which are naturally most similar to the input. Therefore, qualitatively we can observe that using the movieTagline and movieOverview as the features for similarity, we can get high quality results.

2) movieGenre + movieCast

The features used here are: Genre keywords(all the different genres that apply to the movie), cast(the names of the whole cast).

```
1 get_recommendations2('Spider-Man').head(10)

5103                                    Hidalgo
1069                      The Amityville Curse
7353    Cirque du Freak: The Vampire's Assistant
2680                    Defending Your Life
7401                                Pontypool
6695    The Librarian: Return to King Solomon's Mines
1262                      Picture Perfect
8851                    The Age of Adaline
4338                                Scanners
3270                            Cry Freedom
Name: title, dtype: object

[182] 1 get_recommendations2('The Dark Knight').head(10)

3108                          The Way of the Gun
4329                                     Fingers
1010                                           M
8026    Across the Line: The Exodus of Charlie Wright
7274          Gangster's Paradise: Jerusalema
6797                                 Elite Squad
7757                                  Win Win
8401                              Machete Kills
6773                              Lust, Caution
1822                                   Holy Man
Name: title, dtype: object
```

Fig. 6: Top 10 similar movies

The results obtained are very low quality and not similar at all. The reason is that using the list of the whole cast disturbs the generalization and the values obtained in the similarity matrix are also very low.

3) Movie MetaData

Using the MetaData(Director, Genre, Cast, Keywords), the results obtained are more generalized and practical. Here, only the top 3 names from the cast are used so that the results are not bad.

In the Fig.7 and Fig.8 the recommendations obtained are a mix of the sequel movies, movies from the same director and movies with similar cast as well.

```
1 get_recommendations('The Dark Knight').head(10)

8031         The Dark Knight Rises
6218                Batman Begins
6623               The Prestige
2085                  Following
7648                  Inception
4145                   Insomnia
3381                    Memento
8613                Interstellar
7659    Batman: Under the Red Hood
1134                Batman Returns
Name: title, dtype: object
```

Fig. 7: Top 10 similar movies

```
1 get_recommendations('Spider-Man').head(10)

6757                 Spider-Man 3
5538                 Spider-Man 2
271          The Quick and the Dead
8370    Oz: The Great and Powerful
7306                Drag Me to Hell
3237                       The Gift
2307          For Love of the Game
1032                  Evil Dead II
1918                A Simple Plan
4947                       Darkman
Name: title, dtype: object
```

Fig. 8: Top 10 similar movies

All of these approaches lacked one thing, the movies with low ratings were also recommended just because they passed the threshold of similarity with the given movie. This is an issue since, other movies which are not as similar as this one but have a higher rating are ignored. To solve this issue, the IMDB's weighted rating formula was used to sort the list of similar movies by ratings.

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

The result we got after using this formula and sorting the movies is by far the best result we have had, considering that it recommends the similar and highest rated movies simultaneously.

```
In [62]: ImprovedRecommendations('The Dark Knight')
```

Out[62]:

	title	vote_count	vote_average	year	wr
7648	Inception	14075	8	2010	7.919065
8613	Interstellar	11187	8	2014	7.898936
6623	The Prestige	4510	8	2006	7.762198
3381	Memento	4168	8	2000	7.744491
8031	The Dark Knight Rises	9263	7	2012	6.922734
6218	Batman Begins	7511	7	2005	6.905676
1134	Batman Returns	1706	6	1992	5.848168
132	Batman Forever	1529	5	1995	5.051917
9024	Batman v Superman: Dawn of Justice	7189	5	2016	5.013324
1260	Batman & Robin	1447	4	1997	4.281221

Fig. 9: Top 10 similar movies

B. Gaussian Mixture Model with Expectation Maximization

The Gaussian distributions were initialized with random mean and variances with equal weights. In the following figure we can observe how the distribution fit to the given datasets after 50 iterations of the Expectation Maximization.

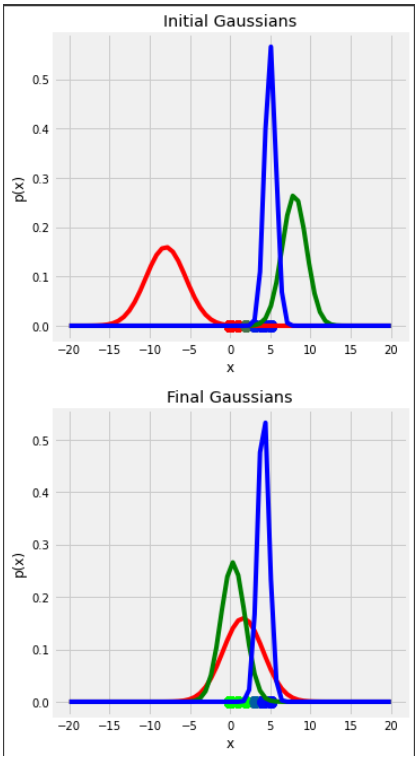

Fig. 10: Gaussain distributions visualized

After fitting the distributions to the given data-points, the responsibility matrix is recalculated and the predicted values are assigned as described in the implementation.

The following figure shows the original rating matrix alongside the predicted rating matrix. You can see that some of the ratings which were zero in the original matrix are now assigned some values, these are the predicted values. This matrix and RMSE were obtained while using 2 Gaussian distributions.

```
Original Matrix:
[[5. 3. 4. ... 0. 0. 0.]
 [4. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [5. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 5. 0. ... 0. 0. 0.]]
Predicted Matrix:
[[5. 3. 4. ... 0. 0. 0.]
 [4. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 ...
 [5. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [3. 5. 1. ... 0. 0. 0.]]
0.7147657346241669
```

Fig. 11: RMSE vs number of clusters

The following image shows the RMSE values for different number of distributions ranging from 1 to 30.

```
array([0.        , 0.7857215 , 0.71476573, 0.70016537, 0.68379831,
       0.68040474, 0.67541847, 0.66518173, 0.66286884, 0.66229983,
       0.65664451, 0.65576251, 0.65355135, 0.65324355, 0.65311372,
       0.64932882, 0.64998437, 0.6494725 , 0.64732167, 0.64423562,
       0.64327536, 0.64203533, 0.64102801, 0.63749579, 0.63630593,
       0.6350518 , 0.63632723, 0.63563975, 0.63225194, 0.63015889,
       0.62883185])
```

Fig. 12: RMSE values

The following is the graph that shows the decrease in RMSE with the increase in number of clusters.

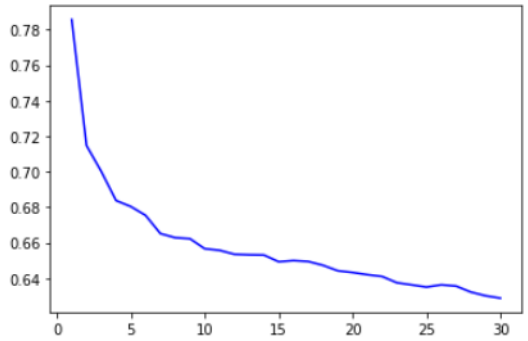

Fig. 13: RMSE vs number of clusters

V. Conclusions

The content based cosine similarity approach is an effective solution to the recommendation problem. The final resulting recommendation list obtained for the input movie is qualitatively feasible and practical as well. However, one of the major drawbacks of this model is that the recommendations would be the same for all the users.

The collaborative filtering method using Gaussian Mixture Models with EM is an interesting approach. The results obtained are purely based on user clustering and ratings and hence the performance is not very accurate.

Based on the results obtained from the approaches implemented in this report, we can infer the following: A hybrid approach using the combination of both content based and collaborative filtering would be the most suitable choice for a real-life implementation of the Movie Recommendation System.

References

[1] Gaussian mixture models — clustering algorithm python. https://www.analyticsvidhya.com/blog/2019/10/gaussian-mixture-models-clustering/. (Accessed on 04/11/2021).
[2] Movielens 100k dataset, Mar 2021.
[3] Movielens 25m dataset, 2021.
[4] Rui Chen, Qingyi Hua, Quanli Gao, and Ying Xing. A hybrid recommender system for gaussian mixture model and enhanced social matrix factorization technology based on multiple interests, Oct 2018.
[5] Anand Rajaraman and Jeffrey David Ullman. *Data Mining*, page 1–17. Cambridge University Press, 2011.
[6] Sandvine. Global internet phenomena.
[7] Ramni Singh, Sargam Maurya, Tanisha Tripathi, Tushar Narula, and Gaurav Srivastav. Movie recommendation system using cosine similarity and knn. pages 2249–8958, 06 2020.