MOVIE RECOMMENDATION SYSTEM AND REVIEW SENTIMENT ANALYSIS USING COSINE SIMILARITY AND LSTM RNN

GUAIDENCE BY: -

DR. T. VETRISELVI

SUBMITTED BY: -

20BCE2063 - ASHWIN BALAJI

20BDS0363 - MUTHUKRISHNAN P

20BDS0364 - SANTHOSHKUMAR S

20BDS0372 - MUGUNDHAN P

ABSTRACT: -

The dataset we will be using for the recommendation system is Kaggle. It has two processes. The first one is demographic filtering which is a general recommendation, for example it shows the popular or most viewed movies. The second one is content based filtering where cosine similarity is used with TF IDF or count vectorizer.

The dataset we used for the review sentiment analysis is the 50K review dataset which has 25K positive and 25K negative reviews. The stemming process is used here, where the root word is found out. Next one hot encoding is used to convert text to vectors(numbers). Word embedding is used to normalize the size of the words using padding. Now LSTM RNN is used to differentiate the positive and negative reviews.

KEYWORDS: -

MOVIE RECOMMENDATION SYSTEM, COSINE SIMLARITY, TD IDF, LSTM RNN, etc.

INTRODUCTION: -

The Basic Concept Behind movie recommendation system is quite simple. The goal of movie recommendation system is to filter and predict only those movies that a Corresponding user is most likely to want to watch. In this Project we are Using NLP technique to pro-process the data such as TF-IDF, CONSIN SIMLARITY, COUNT VECTORIZER. And for sentiment analysis we are using TOKENIZATION, LEMMATIZATION, WORD EMBEDDINGS, LSTM RNN. with the help of these technique and algorithm we are training dataset and testing to predict the Accuracy of the Model.

LITERATURE REVIEW: -

1.A PERSONALIZED MOVIE RECOMMENDATION SYSTEM BASED ON LSTM-CNN:

This project introduces a recommendation algorithm based on LSTM-CNN and applies it to the recommendation of movies by mining user behavior data and recommending movies with higher ratings to them is testing set and training set that the data is divided into, and Top-N recommendation list is produced for the training set, while the algorithm is evaluated on the testing set. In current era, faced with exponentially increasing information, what really confuses people is no longer how to obtain information, but how to quickly extract useful information from massive information. With this dilemma, recommendation system came into being and became an important tool for connecting users and data. A successful recommendation system not only accurately predicts user behavior based on data, but also discovers users 'potential interests, helping them find things that are not easy to find but meet their preferences. In fact, in daily life, recommendation systems are already everywhere. it uses the data provided by the movie website MovieLens. It is testing set and training set that the data is divided into, and Top-N recommendation list is produced for the training set, while the algorithm is evaluated on the testing set. It is the features of the data that LSTM-CNN can effectively extract and complete the recommendation from the results.

2. A HYBRID MOVIE RECOMMENDER SYSTEM BASED ON NEURAL NETWORKS

This paper describes how a combination of the results of content-based and collaborative filtering techniques is used in this work to construct a system that provides more precise recommendations concerning movies. We are not that far away from the time when the sector of entertainment of a "smart home" will be a whole department of its own. For instance, music, plays, books, cinema, films to buy/rent, concerts that suit the preferences of each individual will be successfully suggested. As far as movie recommendations are concerned, the problem of selecting a nice film will get more and more intense as time passes. The Hollywood will eventually come up and anyone will be given the opportunity to be the director of its own virtual movie. To address this problem, we have developed a hybrid system that takes into consideration the kinds of a movie, the synopsis, the participants (actors, directors, scriptwriters) and the opinion of other users as well. Content-based filtering recommends elements to the user based on the descriptions of previously

evaluated items. Therefore, it recommends elements because they are similar to the items that the user has liked in the and solution (Refer Introduction) (5-8 lines) past. User profiles are created by extracting the characteristic features from these evaluated products or services. Such a system has various disadvantages, though. Collaborative filtering matches persons with similar interests and provides recommendations based on this matching. Instead of calculating the similarity between elements, the similarity between users is calculated. The user profile consists of information provided by the user. This information is compared to that provided by other users in order to find the overlaps of interests among individuals There is always the chance of encountering users with particular preferences that are difficult to satisfy. If the number of users is actually small, the quality is low Hybrid systems take advantage of content-based and collaborative filtering characteristics, as the two approaches are proved to be almost complementary. They come through all the constraints described above and as a result they enhance both performance and reliability.

3. COMPARING COLLABORATIVE FILTERING AND HYBRID BASED APPROACHES FOR MOVIE RECOMMENDATION

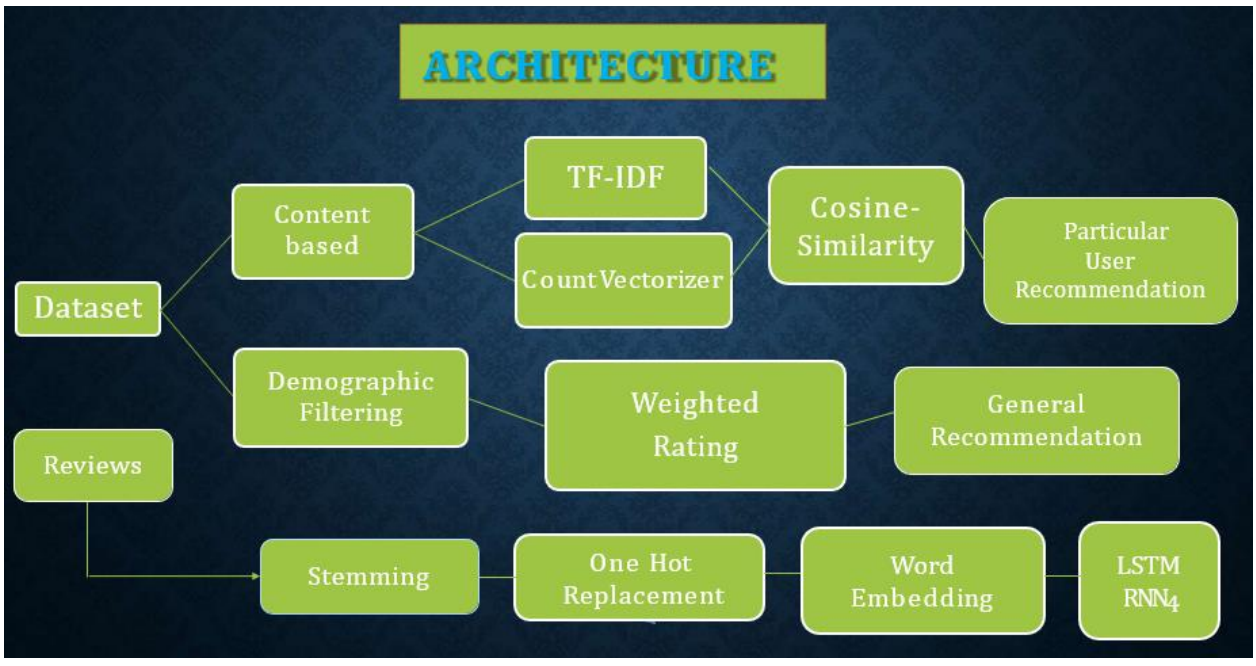
The paper compares the performance of the Collaborative Filtering and Hybrid based approaches in generating movie recommendations. The performance comparisons show that the Collaborative Filtering based approach always outperforms the Hybrid based at any top-N position in Precision and NDCG metrics. These findings conjecture that the Hybrid approach does not always improve the Collaborative Filtering approach in movie recommendation. The Collaborative Filtering approach uses the rating data consisting of two main phases: movie similarity and movie rating prediction. Meanwhile, the Hybrid based approach adds the benefit of a Content-based to the Collaborative Filtering based approach. Thus, it uses both the rating and movie data and is consisting of four main phases: text preprocessing, term weighting, movie clustering, and Collaborative Filtering based approach. The Hybrid based approach is an approach that adds the benefit of a Content-based approach to the Collaborative Filtering based approach [5, 14]. Thus, this approach also requires movie data and additional processes than that of the Collaborative Filtering approach. The Hybrid based approach consists of four main phases to generate the movie recommendations: text preprocessing, term weighting, movie clustering, and Collaborative Filtering based approach.

4. AN IMPROVED COLLABORATIVE MOVIE RECOMMENDATION SYSTEM USING COMPUTATIONAL INTELLIGENCE

The experiment results on Movielens dataset indicate that the proposed approach can provide high performance in terms of accuracy and generate more reliable and personalized movie recommendations when compared with the existing methods. For media product, online collaborative movie recommendations make attempts to assist users to access their preferred movies by capturing precisely similar neighbors among users or movies from their historical common ratings. However, due to the data sparsely, neighbor selecting is getting more difficult with the fast increasing of movies and users. In this paper, a hybrid model-based movie recommendation system which utilizes the improved K-means clustering coupled with genetic algorithms (GA) to partition transformed user space is proposed. The sparse of user-item rating matrix makes it hard to find real neighbors to form the final recommendation list. In our experiments, we compare the performances and some trends of the existing baseline CF movie recommendation systems with our approach, while the neighborhood size varies from 5-60 in an increment of 5. the experiment also demonstrated that our proposed approach is capable of generating effective estimation of movie ratings for new users via traditional movie recommendation systems.

PROPOSED MODEL: -

ARCHITECTURE: -



EXPLANATION: -

In this movie recommendation system, we consider two types of recommendations Demographic filtering and Content-based filtering. Demographic filtering is used for general recommendation for all users, content-based filtering is used for personalized individual user recommendations. To find the high popularity and priority movies related to users' searches, we use some methodologies, let's see those techniques.

1.DEMOGRAPHIC FILTERING: -

This technique is used to provide generalized recommendation for all users in a front page of the recommender system. This recommends a movie based on its popularity and how the movie is being liked by all users. To sort the most popular movies, we use a numeric value which is Weighted Rating (WR) for all movies. The formula to calculate the Weighted Rating is

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

Where,

v – the no. of voters for the movie

m – the minimum votes required to be considered

R – average rating of the movie

C – mean vote across whole report

The higher WR value will have the higher popularity. The system recommends generalized movies based on the WR value to all users.

2.CONTENT-BAESD FILTERING: -

It recommends individual recommendations based on their search. To recommend similar movies, the system uses some features of the movie like actors, director, genre of the movie etc. Here we use two techniques, TF-IDF vectorizer and Count Vectorizer both takes different features of the movie and recommends similar movies. In both cases , we use Cosine Similarity to find similarity score between movies.

TF-IDF VECTORIZER: -

TF-IDF stands for Term Frequency- Inverse Document Frequency. It is a NLP technique, converts text form to numbers, it is easy to process with computer algorithms. It find the most significant word in the document. If the tf-idf value is higher, the word is most important in the document. If the tf-idf value is lower, then the word is less important.

TF-IDF = TF * IDF

TF = No. of times a particular word appears in a sentence / Total no. of words in a sentence

It denotes how frequency, a particular word appears in sentence.

IDF = log(Total no., of sentences / No. of sentences with a particular word)

Here, we are trying to find rare words. Because, in a sentence, stop words are less important since it appears many times in a sentence.

COSINE SIMILARITY: -

It is a technique used to calculate pairwise similarity score for all movies and recommends a movie based on that similarity score.

Cosine Similarity = $\cos(\theta) = A.B / ||A||.||B||$

COUNT VECTORIZER: -

here, We are going to build a recommender based on the following metadata: the 3 top actors, the director, related genres and the movie plot keywords. We need to extract these information from the features in the dataset. The next step would be to convert the names and keyword instances into lowercase and strip all the spaces between them. Then apply count vectorizer method. This will count the No. of occurrences of each word in all overviews and form a matrix. Using this matrix, we can compute similar score for each movie using cosine similarity. This recommender system has been successful in capturing more information due to more metadata and has given us (arguably) better recommendations.

REVIEW ANALYSIS: -

Here, to perform sentiment analyze on reviews we use LSTM-RNN technique.

LSTM-RNN: -

LSTMs enable RNNs to remember inputs over a long period of time. This is because LSTMs contain information in a memory, much like the memory of a computer. The LSTM can read, write and delete information from its memory. This memory can be seen as a gated cell, with gated meaning the cell decides whether or not to store or delete information (i.e., if it opens the gates or not), based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. This simply means that it learns over time what information is important and what is not. In a long short-term memory cell you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn't important (forget gate), or let it impact the output at the current timestep (output gate).

LSTMs enable RNNs to remember inputs over a long period of time. This is because LSTMs contain information in a memory, much like the memory of a computer. The LSTM can read, write and delete information from its memory. This memory can be seen as a gated cell, with gated meaning the cell decides whether or not to store or delete information (i.e., if it opens the gates or not), based on the importance it assigns to the information. The assigning of importance happens through weights, which are also learned by the algorithm. This simply means that it learns over time what information is important and what is not. In a long short-term memory cell you have three gates: input, forget and output gate. These gates determine whether or not to let new input in (input gate), delete the information because it isn't important (forget gate), or let it impact the output at the current timestep (output gate).

EVALUATION: -

According to many research, we come to know that for evaluating performance of a movie recommendation system only accuracy is not the criteria but we should also consider other important parameters. Recommendation quality is also an equally significant metric. 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. It is often used to measure document similarity in text analysis. Distance measures are the fundamental principle for classification, like cosine similarity, which measures the similarity between given data samples. Additionally, choosing a distance metric would have a strong influence on the performance of the classifier. Therefore, the way you compute distances between the objects will play a crucial role in the classifier algorithm's performance. And in review sentiment

analysis we are using the accuracy, recall, precision and F1 score as the metrics it will tell how our model predicted the results.

DESCRIPTION OF DATASET: -

A dataset is a collection of data with a defined structure. That usually presented in tabular form. It contains rows and columns. each column corresponds to a particular variable and each row corresponding to given member of dataset in question.

In this Project We're using Two Different Types of Datasets

One is **TMDB 5000 MOVIE DATASET and another one is 50K Movie Reviews.**

TMDB 5000 MOVIE DATASET: -

The above dataset contains approximately (~) 5000 Movie records. and having Two different csv files.

1) tmdb_5000_credit.csv
2) movies.csv

1) tmdb_5000_credit.csv: -

Feature (movie_id, title, cast, crew).

2) movies.csv: -

Feature (budget, genres, homepage, id, keywords original_language, original_title, overview, popularity, production_company).

50K Movie Reviews: -

The IMDB dataset has 50K movie reviews for text analytics or natural language processing. Which helps to predict the number of positive as well as negative reviews.

IMDB Dataset.csv: -

Feature (review, sentiment)

REFERENCES:

https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie- reviews/metadata

METRICS: -

ACCURACY: -

Accuracy simply measures how often the classifier correctly predicts. We can define accuracy as the ratio of the number of correct predictions and the total number of predictions. When any model gives an accuracy rate of 99%, you might think that model is performing very well but this is not always true and can be misleading in some situations. I am going to explain this with the help of an example.

CONFUSION MATRIX: -

Confusion Matrix is a performance measurement for the machine learning classification problems where the output can be two or more classes. It is a table with combinations of predicted and actual values. A confusion matrix is defined as the table that is often used to describe the performance of a classification model on a set of the test data for which the true values are known. It is extremely useful for measuring the Recall, Precision, Accuracy, and AUC-ROC curves.

PRECISION: -

Precision explains how many of the correctly predicted cases actually turned out to be positive. Precision is useful in the cases where False Positive is a higher concern than False Negatives. The importance of Precision is in music or video recommendation systems, e-commerce websites, etc. where wrong results could lead to customer churn, and this could be harmful to the business. Precision for a label is defined as the number of true positives divided by the number of predicted positives.

RECALL (SENSITIVITY): -

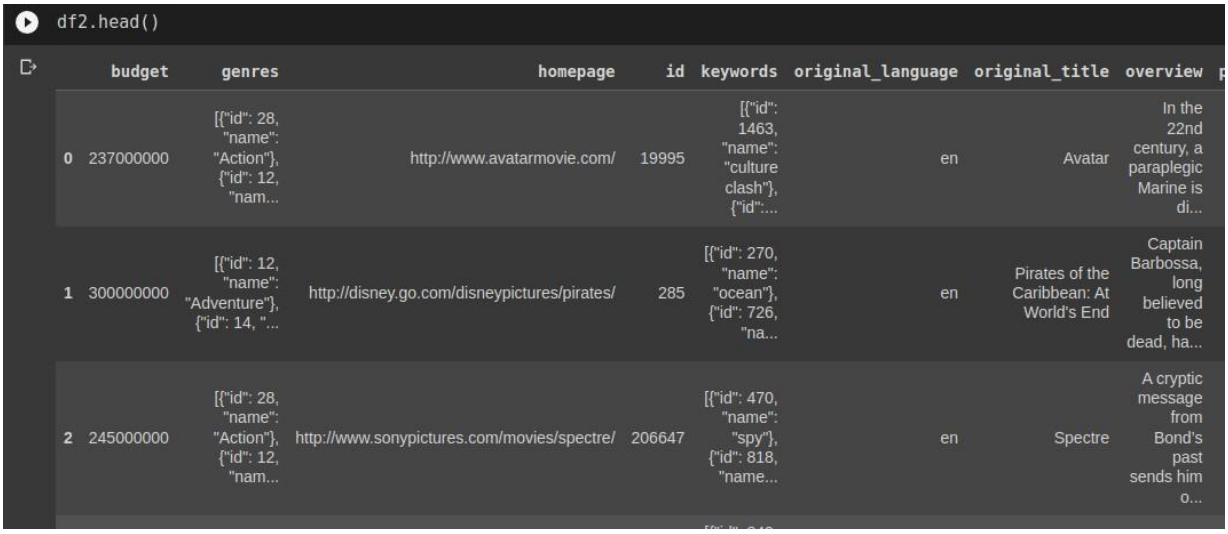
Recall explains how many of the actual positive cases we were able to predict correctly with our model. It is a useful metric in cases where False Negative is of higher concern than False Positive. It is important in medical cases where it doesn't matter whether we raise a false alarm, but the actual positive cases should not go undetected! Recall for a label is defined as the number of true positives divided by the total number of actual positives.

F1 SCORE: -

It gives a combined idea about Precision and Recall metrics. It is maximum when Precision is equal to Recall. F1 Score is the harmonic mean of precision and recall. The F1 score punishes extreme values more. F1 Score could be an effective evaluation metric in the following cases: When FP and FN are equally costly. Adding more data doesn't effectively change the outcome. True Negative is high.

MODEL WITH DATASET: -

DATASET, MODEL AND OUTPUT FOR MOVIE RECOMMENDATION: -



```
df2.head()
```

	budget	genres	homepage	id	keywords	original_language	original_title	overview	p
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	

```python
# Function that takes in movie title as input and outputs most similar movies
def recommend(title, cosine_sim=cosine_sim):
    # Get the index of the movie that matches the title
    idx = indices[title]

    # Get the pairwsie similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    # Return the top 10 most similar movies
    return df2['title'].iloc[movie_indices]
```

```python
recommend('The Dark Knight Rises')
```

```
65                            The Dark Knight
299                            Batman Forever
428                            Batman Returns
1359                                   Batman
3854    Batman: The Dark Knight Returns, Part 2
119                             Batman Begins
2507                                Slow Burn
9           Batman v Superman: Dark of Justice
1181                                     JFK
210                            Batman & Robin
Name: title, dtype: object
```

DATASET, MODEL AND OUTPUT FOR MOVIE REVIEW SENTIMENT ANALYSIS:-

```python
[ ] df = pd.read_csv('/content/drive/MyDrive/IMDB Dataset.csv')
```

```python
[ ] df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

```
embedding_vector_features=400
model=Sequential()
model.add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model.add(Bidirectional(LSTM(100)))
model.add(Dropout(0.3))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 400)          6000000

 bidirectional (Bidirectiona  (None, 200)              400800
 l)

 dropout (Dropout)           (None, 200)               0

 dense (Dense)               (None, 1)                 201

=================================================================
Total params: 6,401,001
Trainable params: 6,401,001
Non-trainable params: 0
```

```
[ ] from sklearn.metrics import confusion_matrix
    confusion_matrix(y_test,y_pred)
```

```
array([[6902, 1306],
       [ 831, 7461]])
```

```
[ ] from sklearn.metrics import accuracy_score
    accuracy_score(y_test,y_pred)
```

```
0.8704848484848485
```

```
[ ] from sklearn.metrics import classification_report
    print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.84      0.87      8208
           1       0.85      0.90      0.87      8292

    accuracy                           0.87     16500
   macro avg       0.87      0.87      0.87     16500
weighted avg       0.87      0.87      0.87     16500
```

COMPARE TABLE: -

AUTHORS	TITLE OF THE PAPER	METHODOLOGY/ ALGORITHM	KEYWORDS	FEATURES
Ramni Harbir Singh, Sargam Maurya, Tanisha Tripathi, Tushar Narula, Gaurav Srivastav (2020)	Movie Recommendation System Using Cosine Similarity and KNN	Content based filtering and KNN	Recommendation System, Content-Based Recommender System, Deep learning.	Accuracy increased by using Content based filtering
Nicolas Hug (2020)	A Python library for recommender systems	Surprise is a Python library for building and analyzing rating prediction algorithms	Surprise library, Recommender system, SVD	Surprise contains model evaluation like cross-validation iterators and built-in metrics as well as tools for model selection and automatic hyper-parameter search
Yeole Madhavi B, Rokade Monika D, Khatal Sunil S (2021)	Movie Recommendation System Using Content Based Filtering	Content based filtering – Count Vectorizer with cosine similarity and Tf-idf Vectorizer cosine similarity	Movie Recommendations, Content-based Filtering, Text to vector, Vector similarity, Hybrid approach	Accuracy Slightly increased by using content-based filtering

Amogh Singhal, Ayushi Khatri, Romyani Saha	MOVIE RECOMMENDATION SYSTEM	K-means Clustering and filtering.	Movie recommendations, hybrid recommender, k-means, Machine Learning.	Similarity and Classification are used to get better recommendations thus increasing precision and accuracy.
Sonu Airen, Jitendra Agrawal **(2021)**	Movie Recommender System Using K-Nearest Neighbors Variants	KNN based collaborative filtering	Accuracy metrics Collaborative filtering K nearest neighbour Machine learning Recommender system	Performance of proposed system can be improved by using another algorithm ex. SVM, ANN, etc.,

CONCLUSION: -

Recommender systems make easiest to users to find the Items what they want exactly. In this project We presented and evaluated algorithms content based filtering and demographic filtering- based recommender system. Content based filtering is for Particular user Recommendation and demographic filtering is for General Recommendation. Our results show that accuracy of **87 percentage.** For review sentiment analysis we used stemming and word embeddings to predict the positive as well as negative reviews and process the data to the deep learning model like LSTM RNN. Furthermore, to increase the accuracy of prediction needs to add more process to recommender system and sentiment analysis.

REVIEW VIDEO LINK: -

https://youtu.be/34lN6qDDSZg