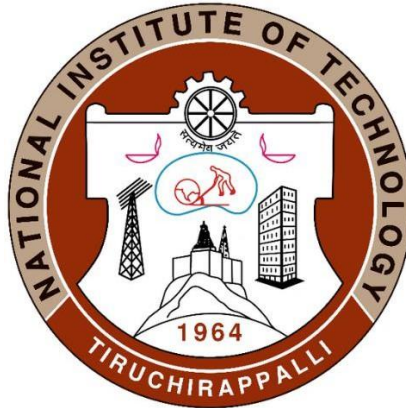


NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI



CSPC 54

INTRODUCTION TO ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

PROJECT REPORT

FACULTY: - Dr. K. Sitara

Team Members: -

Aashish 106121003

Aditi 106121007

Aman 106121011

Animesh 106121013

Sidharth 106121123

MOVIE RECOMMENDATION SYSTEM USING PYTHON

INTRODUCTION

Movie recommendation system is a collaborative filtering or content-based recommendation technique where movie titles are recommended to users based on their choices. Content Based Recommendation System: It uses attributes such as genre, director, description, actors, for movies, to make suggestions for the users. The intuition behind this sort of recommendation system is that if a user liked a particular movie or show, he/she might like a movie like it.

PROBLEM STATEMENT

This recommendation system recommends different types of movies to people. sometime useless for the users who have different taste from the recommendations shown by the system as every user may have different tastes. These systems work on individual users to explore more in recommendation system. Also, many of the users recommends on the basis of Actors, Movies, genres etc.

INPUT

Depending on the models the input here isn't directly provided explicitly by the user rather they are accessed implicitly from the user's perspective and his/her watchlist.

Implicit Input data for this project are:

1. Movie:

- It gives our model a brief idea of the movies and consequently the types of movies the user might be interested in.
- Using the datasets, the model can implicitly find the genre, actor, director of the movie which will be used to recommend similar movies.

2. Actor:

- It's a well-known fact that a user interested in a particular actor's movie finds more interest in that actor as compared to other actors.
- This idea has been well implemented in our model by giving maximum weight to the name of the actor while calculating the similarity Matrix and so it will be well reflected in the output.

3. Director:

- As the trend continues a particular director directs movies sharing significant similarities citing a science-fiction director would most probably come up with scientific movies, which concludes that a user having interest in a particular movie would find interest in a movie directed by the same director.

4. Genre:

- This defines the type of movie be it action, romantic, sci-fi.

5. Rating:

- It contains rating given to the movies by different institutions like IMDB.

6. Language:

- Considers the language and the target audience of the movie.

7. Time range:

- It includes the time when the movies were released, and the model will cover a span including the same.

In our model the dataset provided for the training contains all these details about movies, so when a user searches for a particular movie the model will look for all these input variables and will process the data.

OUTPUT

Our model is designed in such a way that the user is asked to enter a movie name explicitly and the model will give the K number of similar movies as entered by the user.

This output is evaluated using the K Nearest Neighbors machine learning model.

In this model the movie entered by the user will be plotted on a 2D space, where movies are represented as a point and distance between the coordinates signifies similarities, and the nearest K movies would be the recommended movies as expected by the model.

DATASHEET


For training this model we have a TMDb 5000-movie dataset. This dataset contains the data of 5000 most Popular Movies in TMDb's Database. It contains many columns such as movie_Id, crew details, budget, description, tags etc. Processing the dataset into a required manner we can make it easier for the model to recommend the movies, given a movie we give as input.

Detail

Compact

Column

10 of 22 columns

TMDb_Id	IMDb_Id	Title	Original_Title	Overview	Genre
Unique movie id as stored in TMDb's Database.	Unique movie id as stored in IMDb's Database.	English Title of the Movie.	Title of the Movie in the Original Language.	A summary of the Movie.	Genres th to.
	9921 unique values	9680 unique values	9737 unique values	9956 unique values	Drama Comedy Other (85)
419704	tt2935510	Ad Astra	Ad Astra	The near future, a time when both hope and hardships drive humanity to look to the stars and beyond....	Drama Fiction
338762	tt1634106	Bloodshot	Bloodshot	After he and his wife are murdered, marine Ray Garrison is resurrected by a team of scientists. Enha...	Action Fiction

Contribution from an Algorithmic Perspective:

1. pre - processing the data:

We have many attributes that aren't required for the recommendation of the movie. So, we eliminate all the columns and keep only the required ones and trim the dataset. Then we combine all the existing columns into tags and convert them into a single string. We stem the string for duplicates and similar words (like go, goes, going) using the natural language processing library in python. We eliminate primitive words (like is, are, they) using the library and now we have the string of required word.

2. Feature selection and Engineering:

We will try to select those features from the dataset which would result in better movie recommendations from an user's perspective. These features may include the Actor's

Name, the Genre of the movie (whether Action or romantic), Ratings, the involved Director Name, the age group of that particular movie, and the release year of the Movie.

3. Machine Learning Algorithms:

The machine learning model used in this project is K Nearest Neighbour where we would recommend the nearest movies on the 2D space plotted. The other models used can be Neural Networks, Bayesian Inference, Linear Regression, and the movie recommendations algorithm based on Sentiments Analysis and LDA.

4. Evaluation metrics:

In this model we combine some attributes such as genre, director, description, actors of all the movies in the dataset and make a separate attribute called tag.

Each movie with its tag value is represented as a vector in the vector space and the Euclidean Distance is calculated among all these vectors. After that we create a matrix of $n \times n$ dimensions where n =number of movies in the dataset and it represents the similarity of one movie with another movie. Similarity is calculated as the inverse of the Euclidean Distance. Now if the user searches for a particular movie the machine will first find the index of that movie in the dataset and then it will sort that particular row in descending order so that the movies with best similarity value is at front and then finally it will display a certain number of movies.

5. Demographics Data:

Demographics Data is a crucial factor in movie recommendations system as the user preferences depends on geographical location as well stating the fact that people from particular location share, it is obvious that a group of people from similar locality share significant similarities in their interest in movies so it becomes a crucial factor in recommending movies. This integration enables our model to capture the localized impact of geographical location in our project.

Literature Survey

1. Movies recommendations system using K Nearest Neighbour model.

Movie recommendation systems have become an integral part of the entertainment industry, helping users discover new films and enhancing their overall viewing experience. One of the widely used techniques for building such systems is the K-Nearest Neighbors (K-NN) algorithm. This paper provides a comprehensive literature survey of movie recommendation systems based on K-NN, discusses their strengths and limitations, and explores recent improvements and innovations in this domain.

Improvements: -

- Researchers have integrated deep learning techniques, such as neural collaborative filtering, into K-NN models to capture complex user-item interactions and improve recommendation accuracy.
- The use of embeddings to represent users and items has gained popularity. Embeddings can be trained to capture latent factors and enhance recommendation quality.
- Incorporating contextual information, such as user location, time of day, or device used, has become a common practice to provide more relevant recommendations.

ALGORITHM:

```
# Step 1: Data Preparation

# Gather movie data and create a user-item matrix.


# Step 2: Recommendation Generation


# For a user, identify their rated movies and predict ratings for unrated movies.

def generate_recommendations(user_ratings, K):

    recommended_movies = []

    for unrated_movie in unrated_movies:

        calculate similarity scores with rated movies
```



```
select top K neighbors  
  
predict rating for unrated_movie based on neighbors  
  
add unrated_movie and predicted rating to recommended_movies  
  
sort recommended_movies by predicted rating and return top recommendations
```

Input:

User's historical movie ratings and preferences.

Movie dataset containing information about various movies, including genres, release year, and user ratings.

Additional contextual information, such as user location, time of day, or device used (if available).

Parameters for the K-Nearest Neighbors (K-NN) algorithm, including the value of K.

Output:

A list of recommended movies for the user, based on their historical preferences and the K-NN model's predictions.

The predicted ratings or scores for the recommended movies, indicating how likely the user is to enjoy each recommended film.

Optionally, the system may provide explanations or reasons for why certain movies are recommended (e.g., because they are similar to movies the user has liked in the past or because they match the user's current context).

2. Linear regression model for movie recommendations system

Movie recommendation systems are vital for enhancing user experience in the entertainment industry. Linear regression models have gained attention as an approach for building recommendation systems. This paper presents a comprehensive literature survey

of movie recommendation systems based on linear regression and explores their strengths, limitations, and recent improvements in the field.

Improvements: -

- To mitigate overfitting, researchers have incorporated regularization techniques such as L1 (Lasso) and L2 (Ridge) regularization into linear regression models, making them more robust and accurate.
- Advancements in feature engineering, including the extraction of latent features from user interactions and the use of deep learning techniques for feature representation, have improved the performance of linear regression models.
- The integration of deep learning architectures, such as neural collaborative filtering, into linear regression-based recommendation systems has gained popularity. These models capture complex user-item interactions, enhancing recommendation quality.

ALGORITHM:

Step 1: Data Preparation

Gather movie data and user ratings.

Step 2: Model Training

Train a Linear Regression model using user ratings and movie features.

Step 3: Recommendation Generation

Predict movie ratings for a user using the trained model.

def generate_recommendations(user_ratings, movie_features):

for unrated_movie in unrated_movies:

predict rating for unrated_movie using the model

sort movies by predicted rating and return top recommendations



Example of usage:

```
user_ratings = {  
    "Movie1": 4.0,  
    "Movie2": 3.5,  
    "Movie3": 5.0  
}
```

Train the model and generate movie recommendations.

```
generate_recommendations(user_ratings, movie_features)
```

Input:

User's historical movie ratings and preferences.

Movie dataset containing information about various movies, including features such as genres, release year, and user ratings.

Additional user-related data, if available, such as demographics or viewing history.

Parameters for the linear regression model, including regularization strength (e.g., L1 or L2 regularization) and feature engineering techniques.

Output:

A predicted rating or score for each movie in the dataset, indicating how likely the user is to enjoy each movie.

A list of recommended movies sorted by their predicted ratings or scores.

Optionally, the system may provide explanations or reasons for why certain movies are recommended based on the learned coefficients of the linear regression model.

3. Bayesian inference model for movie recommendations system

Movie recommendation systems are essential for enhancing user engagement and satisfaction in the entertainment industry. Bayesian inference models have gained prominence as an approach for building personalized recommendation systems. This paper provides a comprehensive literature survey of movie recommendation systems based on Bayesian inference and explores their strengths, limitations, and recent improvements in the field.

- Improvements: - Efforts have been made to develop scalable Bayesian models that can handle large datasets efficiently. Variational inference and Markov Chain Monte Carlo (MCMC) methods have been applied to Bayesian recommendation systems to improve scalability.
- Bayesian models have been enhanced by incorporating contextual information such as user location, time, and device used. This allows for more context-aware recommendations, making them more relevant and engaging.
- To enhance user trust and satisfaction, research has focused on improving the explainability of Bayesian inference models. Efforts have also been made to address fairness concerns in recommendations to ensure that diverse and equitable suggestions are made.

ALGORITHM:


Step 1: Data Preparation

Gather movie data and create a user-item matrix.

Step 2: Bayesian Model Training

Train a Bayesian inference model (e.g., probabilistic matrix factorization) using user ratings and movie features.

Step 3: Recommendation Generation



```
# For a user, identify movies they've rated and predict ratings for unrated movies using the Bayesian model.
```

```
def generate_recommendations(user_ratings, movie_features):  
    for unrated_movie in unrated_movies:  
        predict rating for unrated_movie using the Bayesian model  
    sort movies by predicted rating and return top recommendations
```

```
# Example of usage:
```

```
user_ratings = {  
    "Movie1": 4.0,  
    "Movie2": 3.5,  
    "Movie3": 5.0  
}
```

```
# Train the Bayesian model and generate movie recommendations.
```


```
generate_recommendations(user_ratings, movie_features)
```

Input:

User's historical movie ratings and preferences.

Movie dataset containing information about various movies, including genres, release year, and user ratings.

Additional contextual information, such as user location, time of day, or device used.



Parameters for the Bayesian inference model, including prior distributions, likelihood functions, and inference methods (e.g., Variational Inference, Markov Chain Monte Carlo).

Output:

A personalized probability distribution over movies, indicating the likelihood of a user enjoying each movie.

A list of recommended movies based on the Bayesian inference model's predictions.

Optionally, the system may provide explanations or reasons for why certain movies are recommended based on the probabilistic model's calculator.

4. Neural Networks for movie recommendations system

Movie recommendation systems play a pivotal role in the entertainment industry, enhancing user experience by providing personalized movie suggestions. Neural networks, particularly deep learning models, have gained prominence in building recommendation systems. This paper presents a comprehensive literature survey of movie recommendation systems based on neural networks and explores their strengths, limitations, and recent improvements in the field. Movie recommendation systems play a pivotal role in the entertainment industry, Movie recommendation systems play a pivotal role in the entertainment industry.

Improvements: -

- The introduction of NCF models combined the strengths of collaborative filtering and deep learning. These models learned user and item embeddings through neural networks, capturing both user-item interactions and complex patterns in the data.
- Attention mechanisms were integrated into recommendation models to emphasize relevant information and improve recommendation quality. Transformer-based architectures, known for their attention mechanisms, found applications in recommendation systems.
- Neural networks have been adapted for multi-modal recommendations, incorporating images, audio, and textual data into recommendation models. This approach enables more holistic and engaging movie recommendations.

5. Emotions with knowledge graphs for movie recommendations

Movie recommendation systems aim to enhance user experience by providing personalized movie suggestions. Incorporating emotions into recommendation systems using knowledge graphs has become an emerging trend, allowing for more emotionally resonant recommendations. This paper provides a comprehensive literature survey of movie recommendation systems that integrate emotions with knowledge graphs and explores their strengths, limitations, and recent improvements in the field.

Improvements: -

- Recent research has focused on extracting emotions from various data sources, including user reviews, audiovisual content, and physiological signals. This allows for a more comprehensive understanding of movie emotions.
- The use of Graph Neural Networks (GNNs) has gained popularity in emotion-based knowledge graph recommendation systems. GNNs can propagate emotional information through the graph structure, enabling more accurate and context-aware recommendations.
- Fine-grained emotion modeling has been employed to capture nuanced emotional states, going beyond simple sentiment analysis. This enables recommendations that consider a broader range of emotional preferences.

6. Movie Recommendation Algorithm Based on Sentiment Analysis and LDA

Movie recommendation systems have become integral for enhancing user engagement on streaming platforms by providing personalized movie suggestions. This paper presents a comprehensive literature survey of movie recommendation algorithms that leverage sentiment analysis and Latent Dirichlet Allocation (LDA), exploring their strengths, limitations, and recent improvements in the field.

Improvements: -

- Deep learning models, such as Recurrent Neural Networks (RNNs) and Transformers, have been applied to sentiment analysis, allowing for more nuanced and accurate sentiment detection in user-generated content.
- Contextual sentiment analysis techniques consider the context in which sentiments are expressed, leading to more accurate emotion detection and improved recommendations.
- Dynamic LDA models have been developed to capture evolving topic distributions over time. This is particularly useful for modeling changing movie preferences and content trends.

GitHub Link: -

<https://github.com/sidharthsinghnitt/Movie-Recommendation-System-Using-Python>

Thank You!