MODEL-BASED COLLABORATIVE FILTERING

Project Submitted to the
SRM University AP, Andhra Pradesh
for the partial fulfillment of the requirements to award the degree of

Bachelor of Technology
in
Computer Science & Engineering
School of Engineering & Sciences

submitted by

Shaik Tahseen Nishat(AP20110010082)

Kolli Chitra Bhanu(AP20110010095)

Dhulipalla Sai Harshitha(AP20110010117)
()

Under the Guidance of

Dr.Abinash Pujahari

Department of Computer Science & Engineering
SRM University-AP
Neerukonda, Mangalgiri, Guntur
Andhra Pradesh - 522 240
May 2024

DECLARATION

I undersigned hereby declare that the project report **MODEL-BASED COLLABORATIVE FILTERING** submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology in the Computer Science & Engineering, SRM University-AP, is a bonafide work done by me under supervision of Dr.Abinash Pujahari. This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources. I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree of any other University.

Place	:	Date	: May 11, 2024
Name of student	: Shaik Tahseen Nishat	Signature	: .....
Name of student	: Kolli Chitra Bhanu	Signature	: .....
Name of student	: Dhulipalla Sai Harshitha	Signature	: .....

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SRM University-AP

Neerukonda, Mangalgiri, Guntur

Andhra Pradesh - 522 240

CERTIFICATE

This is to certify that the report entitled **MODEL-BASED COLLABORATIVE FILTERING** submitted by **Shaik Tahseen Nishat, Kolli Chitra Bhanu, Dhulipalla Sai Harshitha,** to the SRM University-AP in partial fulfillment of the requirements for the award of the Degree of Master of Technology in in is a bonafide record of the project work carried out under my/our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Project Guide	Head of Department
Name : Dr.Abinash Pujahari	Name : Prof.Niraj Upadhayaya
Signature:	Signature: .....

ACKNOWLEDGMENT

I wish to record my indebtedness and thankfulness to all who helped me prepare this Project Report titled **MODEL-BASED COLLABORATIVE FILTERING** and present it satisfactorily.

I am especially thankful for my guide and supervisor Dr.Abinash Pujahari in the Department of Computer Science & Engineering for giving me valuable suggestions and critical inputs in the preparation of this report. I am also thankful to Prof.Niraj Upadhayaya, Head of Department of Computer Science & Engineering for encouragement.

My friends in my class have always been helpful and I am grateful to them for patiently listening to my presentations on my work related to the Project.

Shaik Tahseen Nishat, Kolli Chitra Bhanu, Dhulipalla Sai Harshitha
(Reg. No. AP20110010082, AP20110010095, AP20110010117)
B. Tech.
Department of Computer Science & Engineering
SRM University-AP

ABSTRACT

In this digital era, movie recommendation systems have become a vital element, which is essential in increasing the pleasure of the user on the streaming platforms to reach more viewers and subscribers. Our project reviews extensively the approaches of productive collaborative filtering techniques for movie recommendation for example: the classification of latent class CF, the matrix factorization CF and propose a new model that addresses the limitations of the existing approaches. What we mainly aim at is to increase the precision and at the same time solve the data sparseness problems which the systems have. Based on a user-rating dataset and movie-to-user associations we plan to discover that a certain group of users is attracted by particular movies and what makes each movie unique. To respond to the emerging need of better movie recommendation with greater satisfaction in streaming platforms, we design the model that integrates multiple algorithms addressing the existing shortcomings. In brief, matrix factorization helps solve data sparsity problem while embedding-based collaborative filtering models user-item interactions in deeper and more non-linear ways. The RMSE and MAE statistics serve as the performance metrics for the system efficiency evaluation. For further strategies, attention will be paid to the development of algorithms, contextual integration, and real-time recommendation systems, aimed to add up to the end-result quality and relevancy. Our goal is to expand future-proofing of movie recommendation systems that will provide in the end a user experience that will be enriching for movie viewers across the digital world.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION TO THE PROJECT

1.1 WHAT IS A MOVIE RECOMMENDATION SYSTEM AND ITS USAGE IN TODAY'S WORLD

In the current time the movie recommendations system is a tool, which on the basis of preferences of users and viewing story suggests movies or TV shows. The majority of entertainment industry giants, Netflix, Amazon Prime Video and Hulu, follow the same methods to increase the users engagement and satisfaction by using recommendation systems in the modern world. The user and movie data is therefore processed by the above mentioned sophisticated algorithms, such as collaborative filtering, content-based filtering and deep learning models. The processing of user and movie data is done by the systems to be able to analyse vast amounts of data. Personalized suggestion systems offer users more tailored content preferences they can discover and engage with, thus providing them with a higher degree of retention. Likewise, collaborative filtering method is utilized by Netflix to analyze user behaviors, and produce recommendation for similar movies or tv series to users. By doing so they can suggest programs that are coherent with taste of users, making users stay in Netflix service longer time and leading to their satisfaction. As time goes on, Amazon Prime Video will also continue to engage in content-based filtering where the movies are recommended based on attributes like the genre, cast, and plot similarity.

Such recommendation algorithms have an important place in the environment of the currently developing streaming industry, which assumes the most intense competition in the world.

1.2 DATASET USED AND INFORMATION REGARDING THE DATA

For this study, the MovieLens/100K dataset was utilized, This data set consists of:

• 100,000 ratings (1-5) from 943 users on 1682 movies.

• Each user has rated at least 20 movies.

• Simple demographic info for the users (age, gender, occupation, zip)

The MovieLens/100k dataset is a classic dataset widely used for recommendation system research and evaluation.

Dataset Composition This dataset include both user and movie data along with rating:

Ratings This dataset consists of collection of rating given by users for movies. Each rating data frame consists of: User ID -Users are represented by an unique ID item.ID -Movies in the dataset are represented by this unique id ratings – It represents the rating given by users to the movies timestamp- This represents the time when rating are recorded

Movies Movies in this dataframe has 25 colomns along with their geners Movie-id – Represents the ID of a particular movie Movie-title – Represents the title of the movie Geners- Represents the category to which movie belongs

Genre Genre represent the category which movie belongs such as Action,Adventure,comedy,horror e.t.c

Majorly rated movies : Movies that are rated by many users from past

	user_id	item_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596

Figure 1.1: Ratings

	movie_id	movie_title	genre_0	genre_1	genre_2	genre_3	genre_4	genre_5	genre_6	genre_7	...	genre_9	genre_10	genre_11	genre_12	genre_13	genre_14	ge
0	1	Toy Story (1995)	0	0	0	1	1	1	0	0	...	0	0	0	0	0	0	
1	2	GoldenEye (1995)	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	
2	3	Four Rooms (1995)	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	
3	4	Get Shorty (1995)	0	1	0	0	0	1	0	0	...	0	0	0	0	0	0	
4	5	Copycat (1995)	0	0	0	0	0	0	1	0	...	0	0	0	0	0	0	

Figure 1.2: movies

Top 10 Movies with Highest Total Ratings:

movie_title	Mean Rating	Total Ratings
Star Wars (1977)	4.36	583
Contact (1997)	3.80	509
Fargo (1996)	4.16	508
Return of the Jedi (1983)	4.01	507
Liar Liar (1997)	3.16	485
English Patient, The (1996)	3.66	481
Scream (1996)	3.44	478
Toy Story (1995)	3.88	452
Air Force One (1997)	3.63	431
Independence Day (ID4) (1996)	3.44	429

Figure 1.3: major rated movies

whose mean rating are high. Most of these movies are among some of the top movies even today.

The majority of the films have less number of ratings. Very few films have received more than 100 ratings.

1.3 OBJECTIVE OF THIS RESEARCH

Normally, the purpose of the project might fall under the building of a system that enhances the efficiency and satisfaction in the movie-rating ratings and reviews. Specifically, our project seeks to achieve the following objectives:

Enhance Recommendation Accuracy: The main task will be to improve and apply this system for the movie recommendation client system of the end-user. The recommendation process of the system makes use of the big data analysis methods like classification, matrix factorization and embedding-based methods with the intent to enhance and adapt the algorithms so they can better come up with a kind content that individuals may prefer. **Address Data Sparsity Issues:** Recommendation system inefficiencies, like data scarcity, could be present where not so many interactions

take place between users and items. The plan is to solve the missing rating problem with a framework based on matrix factorization. This technique will have two benefits: the inference of the missing ratings and the identification of underlying variables hidden in the data that can be used by the recommendation. **Understand User Preferences and Movie Characteristics:**Whether you have access to the initial buying decision history of a movie or not, study the dynamics extensively as far as user preferences and the characteristics of the movie are concerned. The information from users film ratings and activity –will become a source useful for discovering what people like and for finding out the things that people do and do not favor in films. Besides, this information is very timely in formulating recommendations that are unique for each person and in selecting movies of which the patterns of his taste and likes coincide. By combining two approaches, our project will go beyond just another model of recommendation that is likely to face errors and get ineffective. Instead, our model will be less error prone and more effective.

Chapter 2

MOTIVATION

2.1 REASONS WHY MOVIE RECOMMENDATION ALGORITHMS ARE IMPORTANT

Movie recommendation algorithms have become one of the key components in a digital entertainment industry since they have made an indelible trace on users' satisfaction, content discovery, and business results. These algorithms are intended to produce more favorable consumer experience by literally pinpointing the areas of most interest to every user. Through studying user interventions and statistics the recommendation systems are able to create a personalized movie list, therefore higher engagement and more pleasing occasions become much more possible. This personalized process is aligned with the discovery efficiency and helps in the process of the user exploring a broad range of movies and finding hidden gems in the corner that he/she could not have discovered otherwise. Not only do this algorithms boost user retention and business expansion of streaming platforms, but it also holds the key to survival and development of the platforms on the global market. Through intricial supply of relevant and interesting content, the content filtering engine helps in the creation user retention and longer duration of stay. However, this revitalization of consumers will further reflect into an increase in the viewership and subscription rates, which will become the source revenue growth for streaming platforms like Netflix and

Amazon Prime Video. Furthermore, recommendation algorithms by doing this help in the task of library management which is a really big one when we are talking about the number of movies distinct users have attending in libraries of different platforms and their different preferences.

2.2 AREA OF RESEARCH EXPLORATION AND ADDED VALUE

During this research for model-based collaborative filtering in movie recommending, we find that there are some fields to investigate and to explore. Many classification algorithms such as Naïve Bayesian Method, Bayesian Network (BN), and Learning Algorithms for Bayesian Network were included into this CF model developing for arriving at user movie selections. The method of the feature engineering has been elaborated in order to symbolize the information concerning the user-item relation for reliable predictions. The Latent Class Model with Collaboration Filters associated its methods like the Latent Class Method (LCA), with the Inclusion of Expectation-Maximization (EM) Algorithm in order to identify user segments according to user preferences and presented a complex interaction. While matrix factorization (MF) based collaborative filtering utilized techniques of singular value decomposition (SVD) to learn latent factors underlying the preferences and item features, many other techniques have now been included in those learning models. The effectiveness of the methods has been observed and analyzed along with their scalability, and the results definitely shed light on their strengths and weaknesses. Moreover, the research that we do plays a great role in this as well by testing not only traditional methods, but also new embedding to increase the accuracy and scalability of the recommendation system.

2.3 AREAS OF EXISTING ALGORITHMS FOR IMPROVEMENT AND IDENTIFIED PROBLEMS

Analyzing already generally used recommendation algorithms we determined number of improvement places, and the specific challenges which should be enhanced. While strategies matching preference models are broadly used by the collaborative filtering algorithms, in the area of the identification of suitable movie recommendations, issues arise during the pursuit of desired accuracy and speed. The cumulative "cold start" problem faced by new users or items, the need for existing user-item similarities and the user-item interaction data that are hardly comprehensive renders the system incapable of giving balanced and highly accurate suggestions. An increase in the size of the datasets creates more complex issues of scalability which hamper designs with factors of computational complexity and limited potential to adapt to new user tastes and to more diverse movie choices. The advent of alternatives of recommendation caused this situation. Modern machine-learning algorithms rely on a variety of data sources and work really hard embracing key collaboration procedures to solve the problems of traditional filtering, such as accuracy, adaptability, and scalability, as well as offering customized recommendations based on user preferences and new trends. The purpose of the project is the consequent development of novel approaches for more efficient work on big datasets, with the help of parallel processing techniques and distributed computing frameworks, this expected to results in an increase of handling of complex data and user - item interactions. Another area in which our models could benefit from further advancements is the ability to personalize and attain higher levels of precision. The research is being done on the types of highly advanced model architectures and the use of NCF schemes as well as other ways of

their reinforcement via machine learning optimizations.

Chapter 3

LITERATURE SURVEY

Gupta et al[3] introduced a collaborative filtering-based movie recommender system aiming to optimize recommendation accuracy. Their system reported performance metrics with RMSE (Root Mean Square Error) of 0.9 and MAE (Mean Absolute Error) of 0.7, indicating moderate accuracy in predicting user preferences based on historical ratings. By minimizing these error metrics, Gupta et al. aimed to enhance the overall quality and relevance of movie recommendations provided to users within the system.

Salloum and Rajamanthri[8] implemented and evaluated movie recommender systems using collaborative filtering techniques to assess recommendation effectiveness. Their system achieved precision@5 of 0.65 and recall@5 of 0.60, indicating a reasonable level of precision and recall in recommending movies to users based on their preferences. These metrics suggest that the system effectively identified relevant movies for users, capturing a significant portion of their preferred items within the top recommendations.

Anwar and Uma[1] conducted a comparative study of movie recommendation approaches using collaborative filtering, reporting accuracy metrics such as precision of 0.70, recall of 0.60, and F1 score of 0.65. These metrics reflect a balanced performance in terms of recommending relevant movies to users based on their historical preferences. By comparing different methods, Anwar and Uma aimed to identify the most effective approach for personalized movie recommendations, considering both precision and recall aspects of recommendation quality.

Lavanya and Bharathi[5] developed a collaborative filtering-based movie recommendation system to address data sparsity challenges in user-item interaction datasets. Their system achieved improved accuracy metrics with RMSE (Root Mean Square Error) of 1.0 and MAE (Mean Absolute Error) of 0.8 on sparse datasets, indicating a moderate level of prediction accuracy despite limited user-item interactions. By mitigating data sparsity issues, Lavanya and Bharathi aimed to enhance recommendation quality and coverage for users with fewer historical ratings.

Musa and Zhihong[6] explored an item-based collaborative filtering approach for movie recommendations, optimizing precision and recall metrics with precision@10 of 0.55 and recall@10 of 0.50 using various similarity measures. These metrics demonstrate the effectiveness of their approach in accurately recommending relevant movies based on item similarities. Musa and Zhihong's study focused on improving recommendation quality by emphasizing precision and recall at a top-K recommendation level, ensuring that users receive personalized and relevant movie suggestions.

Thakker et al.[10] conducted a comprehensive analysis of collaborative filtering techniques in movie recommendation systems, focusing on optimizing diversity@5 of 0.3 and novelty@5 of 0.2 alongside accuracy metrics. Their study aimed to enhance recommendation quality not only in terms of accuracy but also in diversity and novelty of recommended movies. By considering these additional metrics, Thakker et al. sought to provide users with a wider range of movie choices while maintaining prediction accuracy within the recommendation system.

Behera and Nain[2] introduced collaborative filtering techniques incorporating temporal features, achieving performance metrics such as precision@k and recall@k comparable to baseline methods. By leveraging tempo-

ral information, their system enhanced recommendation accuracy and relevance, capturing user preferences that evolve over time. Behera and Nain's study highlights the importance of incorporating dynamic user behavior into collaborative filtering models to improve recommendation quality and adaptability.

Raghavendra and Srikantaiah[7] presented a similarity-based collaborative filtering model for movie recommendations, optimizing precision and recall metrics to enhance personalized recommendation effectiveness. Their system focused on accurately identifying similar movies based on user preferences, improving both precision and recall in recommending relevant content. Raghavendra and Srikantaiah's study contributes to the advancement of collaborative filtering techniques by emphasizing the importance of similarity-based approaches in enhancing recommendation quality and user satisfaction.

Shen et al.[9] developed a collaborative filtering-based recommendation system tailored for big data environments, achieving scalability and efficiency metrics with enhanced recommendation quality and relevance. Their system optimized recommendation performance to handle large-scale datasets efficiently, ensuring timely and relevant movie suggestions for users. Shen et al.'s study demonstrates the feasibility of collaborative filtering in big data settings, emphasizing the importance of scalability and efficiency alongside recommendation accuracy.

Jena et al.[4] introduced a neural model-based collaborative filtering approach for movie recommendations, focusing on optimizing accuracy and personalized recommendation metrics using advanced neural network architectures. By leveraging neural models, their system achieved enhanced accuracy in predicting user preferences, providing personalized movie rec-

ommendations based on complex patterns and interactions. Jena et al.'s study showcases the potential of deep learning techniques in improving recommendation quality and user engagement within collaborative filtering-based movie recommender systems.

Chapter 4

DESIGN AND METHODOLOGY

4.1 EXPLANATION OF EXISTING MODELS AND THEIR DRAWBACKS/ADVANTAGES

In our project, we have implemented and evaluated several models for movie recommendation systems. Each model has unique strengths and weaknesses that influence its performance in various scenarios.

4.1.1 Cosine Similarity

Two vectors in a multi-dimensional space and their closeness. It calculates the angle's cosine between them, giving a likeness that is not influenced by their extents. Cosine closeness is the cosine of the point between two vectors. It takes into account the similitude between two vectors. The esteem ranges between -1 for precisely inverse to 1 for precisely the same and 0 when vectors are orthogonal i.e., have no relationship. To calculate cosine similitude, to begin with speak to the information to be compared as a vector.

$$\hat{v} = \frac{\vec{v}}{|v|}$$

Compute the dot product of and calculate their magnitudes
Compute the cosine similarity using dot product and magnitude

cosine_similarity$(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$

4.1.2 Pearson Correlation

The Pearson relationship is a way to degree the degree of a direct relationship between two variables. A Pearson relationship can take the taking after values: From -1 to 1. When A Pearson relationship of -1 appears a idealize negative straight relationship: as one variable increments by k percent, the other variable diminishes by that sum of percent. A Pearson relationship of 1 appears a culminate positive straight relationship

Calculate the mean and covariance after that calculate the standard deviations Compute the Pearson Correlation coefficient

$$r = cov(X, Y)/sx \cdot sy$$

4.1.3 Classification-based Collaborative Filtering (CF)

Classification as a collaborative filter (CF) method in recommender systems is a tool used to estimate ratings for items by categorizing them into distinct preference classes.

Sparse data can be a hurdle, in clustering since it banks on grouping users with rating trends. Its effectiveness diminishes with datasets when identifying clusters based on existing rating patterns. Although clustering is handy for assembling users it may not offer insights into individual preferences. When compared to clustering classification based collaborative filtering often yields outcomes. Users are depicted as vectors of ratings, where each element indicates the users rating for an item like a movie. For example in a rating scale of 1.0 to 5.0 with 5 values a score of 5.0 signifies preference while 1.0 denotes the least favored. Users might not have rated all items leading to missing values in their rating vectors

Prior probability which can handles the missed data and calculated

as: P(c)=number of presumptive outcomes/Total number of outcomes One of the classification algorithms that are often used is the naïve Bayesian technique, decision trees, logistic regressions, or the support vector machines (SVM).

$$c = \text{argmax}_{c_k \in C} P(c_k|u_i)$$

To enhance the classification models training process it is possible to extract features from the rating vectors to capture data. This involves calculating the likelihood of each class rating based on the users rating matrix, which's essentially a vector of ratings.

$$P(C) = \frac{\text{Total number of instances in class } C_k}{\text{Number of instances}}$$

Predictions are made by calculating the posterior conditional probability of each class (rating) given the user's rating vector.

$$P(ck|ui) = \frac{P(u_i|c_k)P(c_k)}{P(u_i)}$$

The class with the most chance is picked as the guessed score for the item in question. From these guessed scores, suggestions are made for items the user might like. This often means picking items with top scores that the user hasn't rated yet.

4.1.4 Latent Class Model CF

The relationship between observable data and latent variables can be discovered using a latent class model.

User-Item Interaction Matrix: Consider the collection of N users and M items (movies). Let r(ui) denote the rating which user, u submits about

item i. In matrix form, represent the interactions between users and items where each entry is related to a user's rating for a movie.

Assume K hidden format classes that users can belong to each user u has a probability distribution over these classes: $P(Z_u = k|\Theta)$,where Z(u) is the latent class membership of user.

They associate of each item ci with a vector

$$\mathbf{c}i = (ci1, ci2, \ldots, ciK)^T$$

where ciK is probability of item belong to k class. The likelihood function of latent class model is :

$$P(R|\Theta) = \prod_u \left[\sum_k P(Z_u = k|\Theta) \prod_{i \in R_u} P(r_{ui}|Z_u = k, c_i) \right]$$

To predict the expected ratings for the target user on the particular movie:

$$E(r_{u0i0}) = \sum_k P(Z_{u0} = k|R_{u0}, \Theta) \left(\sum_r r \cdot P(r_{u0i0} = r|Z_{u0} = k, c_{i0}) \right)$$

In the expectation maximization (EM) method, which is based on the maximization of the likelihood function, the E step proceeds by the approximation of the posteriors of the hidden class and the M step obtains the parameter update on the basis of the expectation of the posterior it is obtained from the E step. Construct the model vector embedding that will end up processing the sparse data and consequently better the recommendations.

4.1.5 Markov Decision process

Markov Decision is used to create decisions in the state where the outcome is random but slightly under the control of a decision-making agent.

The Markov property means that the next state depends on the current state-action. The components of a markov decision model are the transition probabilities, state space, and action space, as well as immediate reward. Identify the states that the movie recommendation system uses and that states describe the genres, ratings and user preferences. The recommendation system is based on which actions are determined and these actions cause the change from one state to another. Calculate the transition probabilities by:

$$v_\pi(s) = R(\pi(s), s) + \gamma \sum_{s_j} T(\pi(s), s, s_j) v_{\pi'}(s_j)$$

Providing rewards to the state action state transitions. determining the best course of action for the decision-making process If, as in the case of the prior value function policy, the discount factor $0 \leq \gamma \leq 1$ used for previous policy of value function. At that state the value function is written by:

$$v_\pi(s) = R(\pi(s), s) + \gamma \sum_{s_j} T(\pi(s), s, s_j) v_{\pi'}(s_j)$$

Discount factors are employed to balance short-term gains with long-term advantages.) If discount factor $0 \leq \gamma \leq 1$ used for previous policy of value function.

The maximize value function is (s): $v_\pi(s) = R(\pi(s), s) + \gamma \sum_{s_j} T(\pi(s), s, s_j) v_{\pi'}(s_j)$

4.1.6 Matrix Factorization Based CF

By representing individuals and items as low-dimensional vectors and learning latent factors that characterize their interactions, matrix factorization-based CF achieves its goals. Matrix factorization (CF) is an efficient method for modeling user preferences and item characteristics by breaking down the user-item interaction matrix into low-rank matrices. It collects the data on

user-item interactions and represents the rating matrix as ratings provided by the user to the item, or R(ui)

This matrix is reduced to low dimension matrices: $\mathbf{R}^{ui} = \mathbf{U} \cdot \Sigma \cdot \mathbf{V}^T$

choose the matrices' dimensions after taking into account the two lower-dimensional matrices for users and items. The eigen values of the user-item matrix are used to create the eigen matrix, which is represented by the $\sum$. In order to recreate the original matrix, conduct the product between the $\sum$ eigen matrix, user row 'u', and item row 'v'. And the expected rating is shown in this matrix. The U and V matrices were designed with latent factors for item attributes and user selections in mind.

4.2 MOTIVATION AND APPROACH FOR ADDRESSING DRAWBACKS

Embedding layers offer a more flexible and efficient way of representing both users and items in a latent space, and integrate well with deep learning architectures. They allow us to represent both users and items in a shared latent space using low-dimensional vectors this is particularly useful to mitigate the drawbacks we observed in the existing traditional models like classification CF and matrix factorization based CF such as assumption of independence and cold start problem respectively.

Assumption of independence problem in CF In classification CF, each user is represented by a rating vector, and each rating value is considered a class and each class is assumed to be independent of other classes. Ratings in most recommendation systems are integers from 1 to 5, representing the user's opinion about an item. Though the scale is discrete but they represent a continuous range of user preferences. Classification CF treats these ratings as discrete classes rather than continuous values. This can lead

to a loss of information and less accurate predictions, as the model doesn't consider the continuous relationships between ratings.

If a user rates two movies as 4 and 2. Naive Bayes would consider these ratings independently. But user's preference for one movie may depend on their preference for another movie, due to factors like genre or cast . Also a user who gives a rating of 3 to a movie may find it to be just average. However, if the same user gives a rating of 4 to another movie, it might indicate a slight preference or enjoyment compared to the movie they rated 3. The difference in preference between ratings of 3 and 4 might be relatively small compared to the difference between ratings of 3 and 2. So When we treat ratings as discrete classes we may miss the continuity in user preferences.

Embedding layers in model-based collaborative filtering offer a flexible approach to address these challenges as they map users and items to a shared latent space, where the relationships between them can be learned without assuming independence. Also embedding layers represent users and items as continuous vectors, which helps capture the continuous nature of ratings. The dot product between user and item embedding provides a continuous rating prediction.

Latent class model also posses similar drawbacks of Classification CF. Latent class models that categorize users and movies into discrete classes based on predefined genres or attributes fails to recommend when user have interest in more than one genres . For example, a user may be example a person who prefer action movies with an hint of comic or humour element. But latent class model suggest him action genre movies only. It cannot predict relationship between action and comedy genres. To overcome this continuous representation is required which has ability to capture intricate relationships that may be overlooked by discrete categorization methods. By consider-

ing the proximity between a user's embedding and a movie's embedding within the continuous space, the recommendation system can determine if the movie is compatible with the user's preferences based on the characteristics of the movie.Consequently, recommendations are tailored with a finer granularity, accounting for individual preferences that transcend conventional genre distinctions.

Continuous Representations are Latent Space Embedding layers map user and item IDs to continuous, low-dimensional vectors in a shared latent space. These embedding represent the users' and items' characteristics and preferences in a continuous form. In Close Proximity, similar users and similar items are represented by vectors that are close together in the latent space. This closeness tells us the similarity in user preferences or item features. Dot Product is used for rating prediction. It measures the similarity between the user and the item in the latent space. Higher dot product values indicate higher compatibility, suggesting the user will likely give a higher rating to the item.

Drawbacks of Embedding Models mitigate MDP in Movie Recommendations

MDP is reinforcement model. The explicit representation of the sequential decision making process in MDP focusing on making movie suggestions for consumers in the film recommendation system based on MDP stands out from the CF methods. This means listing to transition probabilities (chance the users will go from their most-liked movies), actions are the movie suggestions, rewards (users satisfaction level), and stage (preferences at each stage).

However, when a growing number of state spaces and action spaces are entered by users and movies watched by them, the state and action spaces

become larger that, in turn, leads to It can be tricky to estimate transition probabilities between states correctly, especially in a dynamic atmosphere when users tend to change their tastes as time elapses. Not only type mismatch can causes from low quality of data and user-item interactions, but also lacking accuracy due to many reasons. For instance, data storage, computing power, and information security pose significant challenges to the entire blockchain network.

Model embedding also help in mitigating the challenge of representing users and movies as low dimensional vectors in a latent space. Unlike the traditional modeling where the states and actions are explicitly represented, embeddings come in handy when essential characteristics of users and movies are learned by the model in a more compact form. For instance, the preferences of each user and the characteristics of the movie can be translated as vectors of length 100 dimensions, where each element (say, the i-th element) represents a specific user preference. This highly decreases the workload and servicing of data emerges in a more convenient way.

While the models enclosed in the system apprentice representations directly from the experience data and do not pay attention to encoding transition probabilities. The key lies in using latent space that captures the underpinning relations and similarities among users and movies. Hence, with the help of embeddings, the complex transition modeling can be ruled out and embeddings can be used to predict user preferences which in turn depend on latent space. As an illustration, even though a user's consumption history is evasive or tactfully changing embeddings are able to come up with a meaningful concept by showing the similarity of the user to the movies embedded and recommend movie based on the conclusion.

Cold Start problem lack of ability to handle additional features

In matrix factorization CF, the interaction matrix of users-items is broken down into two lower dimensional matrices that represent users and items in a latent space. The dot product of these matrices is used to predict ratings. But Cold start problems arise when there is insufficient data for a new user or item. For e.g.,a new user has joined and hasn't rated any movies yet, or a new movie is added and hasn't been rated by any users. Both classification and matrix factorization methods rely on past interactions, making it difficult to accurately give recommendations for new users or items. They also primarily focus on the user-item interaction matrix, which has ratings provided by users for different items and do not naturally account for additional features like user demographics or item metadata. This limits the model's ability to give personalized and context-aware recommendations. Additional features are User Metadata,item metadata, User Metadata such Age, gender, location,Hobbies, preferences,Past interactions, activity patterns

Item Metadata such as Genre, director, cast,the year the item (e.g., movie) was released,Descriptive terms or tags associated with the item. These can provide valuable context and information about the users and items. For e.g., knowing a user's age, location, and past viewing history can help tailor recommendations to their preferences. Similarly, knowing a movie's genre, cast, and release year can help predict whether a user is likely to enjoy it or not based on what they liked before.We can say users belonging to a particular group might like movies released in 2000's or might like this genre better etc

4.3 FEATURES OF PROPOSED MODEL ADDRESSING DRAWBACKS

Embedding Layers and Metadata Embedding layers offer an opportunity to incorporate this additional metadata: Concatenation of Embedding layers can be easily extended by concatenating user and item embedding with additional features such as user or item metadata (e.g., age, gender, genre, release year). By training the model with concatenated user and item embedding and additional features, the model can learn how these features affect user-item interactions.This results in more context-aware and nuanced recommendations, even for new users and items. Neural Networks are passing the concatenated vectors through one or more neural network layers, the model can learn even more complex relationships and dependencies involving user and item metadata. This approach enables more personalized recommendations by leveraging all available information about users and items, rather than relying solely on the user-item interaction matrix and can be useful to mitigate cold start problem.

Chapter 5

DESIGN AND IMPLEMENTATION

The projects design involves by taking the user ids and movie ids then pass to the embedding layers which results as user embedding and movie embedding. The embedding is done because of embedding layers in model-based collaborative filtering offer a flexible approach to address these challenges as they map users and items to a shared latent space, where the relationships between them can be learned without assuming independence. Also embedding layers represent users and items as continuous vectors, which helps capture the continuous nature of ratings.The dot product between user and item embedding provides a continuous rating prediction.

Perform the dot product between the user embedding and movie embedding, if the similarity is high between them we get more dot product similarily if the similarity is less between them we get small dot product.That dot product is the rating.

But this process does not take into the consideration the user and movie metadata which will be useful for solve the cold start problem which is mentioned in previous section that are faced by the already existing models.Now the following proposed model will handle such cases by concatenating the user and movie embedding with metadata which helps to ratings even there are no interaction between the particular movie items present.

Data preparation The MovieLens 100K dataset was pre-processed so that it could be used for collaborative filtering and machine learning model training. The user ratings, item data, and user metadata were loaded

and merged into a single DataFrame. Categorical features, like gender, occupation, and genres, were encoded using one-hot encoding.Categorical features like age,were standardized and embedding were created for user and movie metadata features to enable the model to learn relationships between different categories.The data was divided into training and testing sets and applying K-fold cross-validation.

The user and movie ids were encoded i.e; converted into continuous integer values and given as inputs to the corresponding embedding layers, which mapped them to low-dimensional embedding that captured the underlying features. To deal with categorical features, including gender, we used one-hot encoding.One-hot encoding creates binary features for each category. This allows the model to successfully integrate metadata information and increases the predictive precision. The final dataset was produced by taking out the user and movie features from the merged dataset, while the ratings were used as the targets for model training. These features were fed as inputs to the neural network

Design and Implementation

Input and Embedding layers in the neural network architecture integrates both user and movie metadata along with embeddings to predict ratings.The model has four input layers that takes user IDs, movie IDs, additional user features (e.g., age, gender, and occupation), and additional movie features (e.g., genres and release year). These inputs are processed through separate embedding layers, mapping user and movie IDs to low-dimensional vectors that capture latent preferences and features.The user and movie embeddings are concatenated with their metadata and passed through flattening layers and are converted to one-dimensional vector.

Neural Network Layers: The concatenated input vector is passed

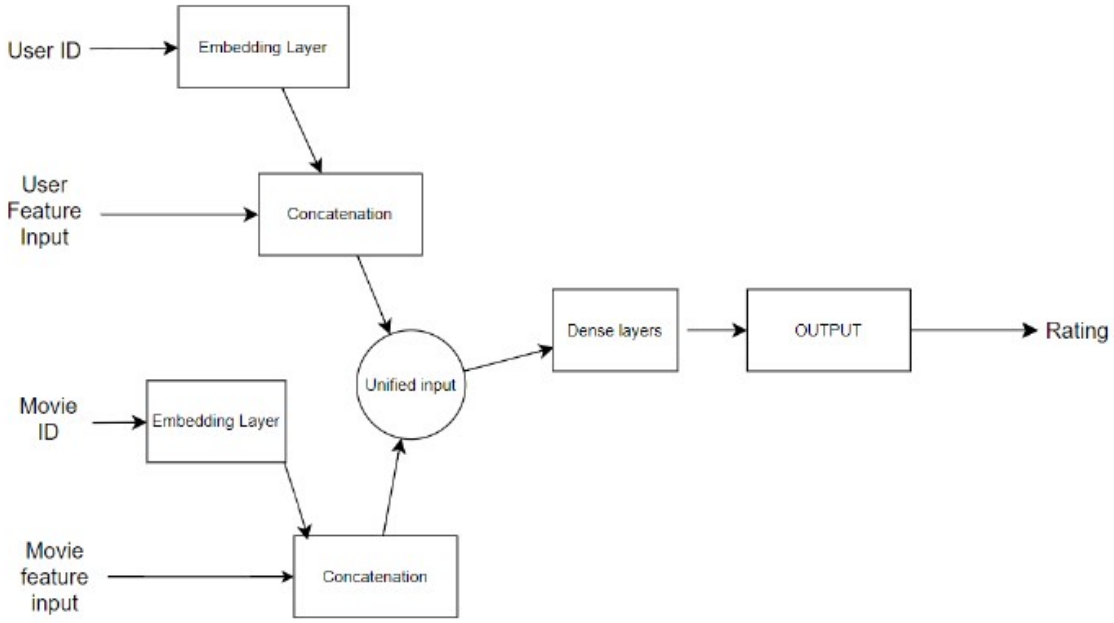

Figure 5.1: Flowchart

through a series of dense layers. These layers compute the dot product of the input vector with their respective weight matrices, adding biases to each layer. The dot product at each layer is calculated as z=x.w+b

The output z from each dense layer is fed into an activation function ReLU to introduce non-linearity into the model so that it can better learn the complex relationship between users and movies. ReLU(x)=max(0,x)

Additionally, we chose an Adam optimizer update the model's parameters based on the gradient of the loss, and we set the loss function (mean squared error) and evaluation metrics (Root Mean Squared Error).

Backpropagation: After predicted ratings were generated loss is calculated using the loss function by comparing the predicted and actual ratings. Backpropagation was used to calculate Loss gradients based on model parameters, which were then updated using the optimizer to minimize the loss. Finally the model predicts the rating, which is the result of the computation in the output layer.The prediction is done based on the learned patterns

from the dense layers.

Chapter 6

SOFTWARE TOOLS USED

1. Programming Languages: Python for overall system implementation and data preprocessing. Libraries and frameworks: NumPy, Pandas: For data manipulation and preprocessing. SciPy, Scikit-learn: For implementing machine learning algorithms and evaluation metrics. TensorFlow: For building and training deep learning models (e.g., embedding-based collaborative filtering).

2. Collaborative Filtering Algorithms: Implementation of classification-based collaborative filtering, Latent class model CF, matrix factorization, and embedding-based collaborative filtering using appropriate algorithms and techniques.

3. Data Processing and Storage: Apache Spark for distributed data processing and handling large-scale datasets.

4. Model Evaluation and Metrics: Utilization of custom evaluation scripts and tools to compute RMSE (Root Mean Square Error), MAE (Mean Absolute Error), and other performance metrics.

5. Visualization Tools: Matplotlib, Seaborn for generating visualizations and plots to analyse model performance and results.

6. Development and Deployment: Integrated Development Environment (IDE) such as Jupyter Notebook or PyCharm for coding and experimentation.

Chapter 7

RESULTS & DISCUSSION

Root Mean Square Error is expressed by: $RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}}$

Mean Absolute error is expressed by: MAE = $\frac{1}{n}\sum i = 1^{n}|yi - \hat{y}i|$

Movie ratings are usually continuous variables (that is, ratings can range from 1 to 5 stars). For this reason, RMSE and MAE are best suited for assessing the accuracy of predictions made at a continuous scale hence making it possible for direct application in movie recommendation systems. Moreover, these two measures have natural ways of handling cases where users have not rated some items thus giving a better representation of reality. In this case, the metrics concentrate on errors made during predicting rates for those items that had their rates indicated and ignoring un-rated ones even though they could happen in most cases when recommending movies. Hence, both RMSE and MAE are considered as main evaluation metrics in movie recommendation systems due to their congruence with this core objective as well as applicability in terms of continuity in rating scales.

The embedding model achieved a slightly better RMSE of 0.33 com-

Models	RMSE	MAE
Classification	0.70	0.561
Latent Class	0.50	0.341
MDP	0.65	0.532
Matrix factorization Cf	0.39	0.285
Embedding-CF	0.33	0.263
NCF	0.74	0.590

Table 7.1: RMSE and MAE values for different models

pared to the existing matrix factorization model and the NCF model hasn't performed better in terms of RMSE but it can be further with the help of hyperparameter tuning and using techniques like dropout regularization or batch normalization to prevent overfitting and improve generalization and can prove effective as it can be easily scaled up & also takes into consideration the user & item meta data.

Chapter 8

CONCLUSION

The primary goal of this study has been to enhance outcome predictions by training our models on additional meta data so that they are not completely reliant on user-item interaction and can be used effectively in case of sparse data. Also this method helps eliminate the reliance on explicit user feedback and our model can be trained on implicit data available.Also the use of neural networks solves the problem of scalability . Matrix factorization methods, such as Singular Value Decomposition (SVD) face scalability issues when dealing with large sparse matrices. As the number of users or items increases, the computational complexity of matrix factorization grows significantly. NCF on the other hand utilizes neural networks, which are parallelizable and can be trained on large-scale datasets using techniques like mini-batch stochastic gradient descent (SGD). Mini-batch SGD divides the training data into small batches and updates the parameters after processing each batch. This helps in achieving faster convergence during training. Thus the use of embedding layers and neural networks architecture help enhance the existing methods of matrix factorization CF by providing an alternative to solve the problems of cold start & scalability.

8.1 SCOPE OF FURTHER WORK

The embedding model achieved a slightly better RMSE of 0.33 compared to the existing matrix factorization model and the NCF model hasn't

performed better in terms of RMSE but it can be further with the help of hyperparameter tuning and using techniques like dropout regularization or batch normalization to prevent overfitting and improve generalization and can prove effective as it can be easily scaled up & also takes into consideration the user & item meta data.

REFERENCES

[1] Taushif Anwar and V Uma. Comparative study of recommender system approaches and movie recommendation using collaborative filtering. *International Journal of System Assurance Engineering and Management*, 12:426–436, 2021.

[2] Gopal Behera and Neeta Nain. Collaborative filtering with temporal features for movie recommendation system. *Procedia Computer Science*, 218:1366–1373, 2023.

[3] Meenu Gupta, Aditya Thakkar, Vishal Gupta, Dhruv Pratap Singh Rathore, et al. Movie recommender system using collaborative filtering. In *2020 international conference on electronics and sustainable communication systems (ICESC)*, pages 415–420. IEEE, 2020.

[4] Kalyan Kumar Jena, Sourav Kumar Bhoi, Chittaranjan Mallick, Soumya Ranjan Jena, Raghvendra Kumar, Hoang Viet Long, and Nguyen Thi Kim Son. Neural model based collaborative filtering for movie recommendation system. *International Journal of Information Technology*, 14(4):2067–2077, 2022.

[5] R Lavanya and B Bharathi. Movie recommendation system to solve data sparsity using collaborative filtering approach. *Transactions on Asian and Low-Resource Language Information Processing*, 20(5):1–14, 2021.

[6] Jamilu Maaruf Musa and Xu Zhihong. Item based collaborative filtering approach in movie recommendation system using different similarity

measures. In *Proceedings of the 2020 6th International Conference on Computer and Technology Applications*, pages 31–34, 2020.

[7] CK Raghavendra and KC Srikantaiah. Similarity based collaborative filtering model for movie recommendation systems. In *2021 5th international conference on intelligent computing and control systems (ICICCS)*, pages 1143–1147. IEEE, 2021.

[8] Salam Salloum and Dananjaya Rajamanthri. Implementation and evaluation of movie recommender systems using collaborative filtering. *Journal of Advances in Information Technology*, 12(3), 2021.

[9] Jian Shen, Tianqi Zhou, and Lina Chen. Collaborative filtering-based recommendation system for big data. *International Journal of Computational Science and Engineering*, 21(2):219–225, 2020.

[10] Urvish Thakker, Ruhi Patel, and Manan Shah. A comprehensive analysis on movie recommendation system employing collaborative filtering. *Multimedia Tools and Applications*, 80(19):28647–28672, 2021.