An

Industry Oriented Mini Project Report

on

HYBRID PRODUCT RECOMMENDATION SYSTEM

A report submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology

by

Krishna Kireeti Morampudi

(20EG105427)

Under the guidance of

Jayendra Kumar

Assistant Professor, CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANURAG UNIVERSITY

VENKATAPUR– 500088

TELANGANA

YEAR 2023-2024

DECLARATION

I hereby declare that the report entitled **"HYBRID PRODUCT RECOMMENDATION SYSTEM"** submitted for the award of Bachelor of technology Degree is my own work and the Report has not formed the basis for the award of any degree, diploma, associateship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma

Place: Anurag University, Hyderabad

Krishna Kireeti Morampudi
20EG105427

CERTIFICATE

This is to certify that the Report / dissertation entitled **"HYBRID RECOMMENDATION SYSTEM"** that is being submitted by **Krishna Kireeti Morampudi (20EG105427)** in partial fulfillment for the award of B.Tech in Computer Science And Engineering to the Anurag University is a record of bonafide work carried out by him under our guidance and supervision.

The results embodied in this Report have not been submitted to any other university or Institute for the award of any degree or diploma

Head of the Department

Signature of theSupervisor
Jayendra Kumar
(Assistant Professor)

ACKNOWLEDGEMENT

I would like to express our sincere thanks and deep sense of gratitude to project supervisor **Jayendra Kumar** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

I would like to express my special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for their encouragement and timely support in our B.Tech program.

I would like to acknowledge my sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Dept. of CSE, Anurag University. I also express my deep sense of gratitude to **Dr. V V S S S Balaram**, Academic Coordinator. **Dr. Pallam Ravi**, Project Coordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated me during the crucial stage of our project work.

ABSTRACT

To enhance the customer experience and to boost up the sales of products, almost all of the companies are trying to make some sort of mechanism that is nothing but a recommendation system. So to finalize this task the recommender system comes into the light. The system works in two steps, first, it analyzes the user search for an item and simply user interests, and secondly, it tries to find a similar set of items that the user may be interested in. This in turn leads to better choices among the products. In today's world, we find a wide variety of search options and we may have difficulty selecting what we really need. The recommendation System plays an important part in dealing with these problems. A recommender system is a framework that is a filtering system that filters the data with various algorithms and recommends the user with the most relevant data.

Recommender Systems commonly face a problem called “Cold Start Problem”. It refers to when items added to the catalog have either none or very little interactions. This constitutes a problem mainly for collaborative filtering algorithms due to the fact that they rely on the item's interactions to make recommendations. This work focuses on using a hybrid approach to overcome this problem

LIST OF FIGURES

LIST OF TABLES

INDEX

1. INTRODUCTION

A recommender system, also known as a recommendation system or engine, is a subclass of information filtering systems that is designed to predict and suggest items or content to users. These systems leverage algorithms and data to make personalized recommendations, such as movies, music, books, products, or other items, to individuals based on their preferences and behaviors.

Recommender systems are commonly used in various online platforms and applications to enhance user experience and engagement. They can be broadly categorized into three main types.

Collaborative filtering methods make recommendations based on the historical preferences and behaviors of users. There are two primary approaches within collaborative filtering. User-Based Collaborative Filtering approach finds users with similar preferences to the target user and recommends items liked by those similar users. Instead of comparing users, the Item-Based Collaborative Filtering approach focuses on finding similarities between items and recommends items that are similar to the ones the user has liked or interacted with.

Content-based filtering recommends items to users based on the attributes and features of the items themselves. It looks at the characteristics of items and matches them to the user's profile of preferences. For example, if a user likes action movies, the system recommends other action movies.

Hybrid systems combine multiple recommendation techniques to provide more accurate and diverse recommendations. For instance, a hybrid system might integrate collaborative filtering and content-based filtering to leverage the strengths of both approaches.

Recommender systems are widely used in e-commerce, social media, streaming services, news websites, and more. They help users discover new products or content, increase user engagement, and often lead to higher user satisfaction and retention.

These systems rely on various data sources and machine learning techniques to continuously improve the quality of recommendations. They can incorporate factors like user ratings, historical behavior, item features, demographic information, and contextual data to tailor recommendations to individual users' tastes and preferences.

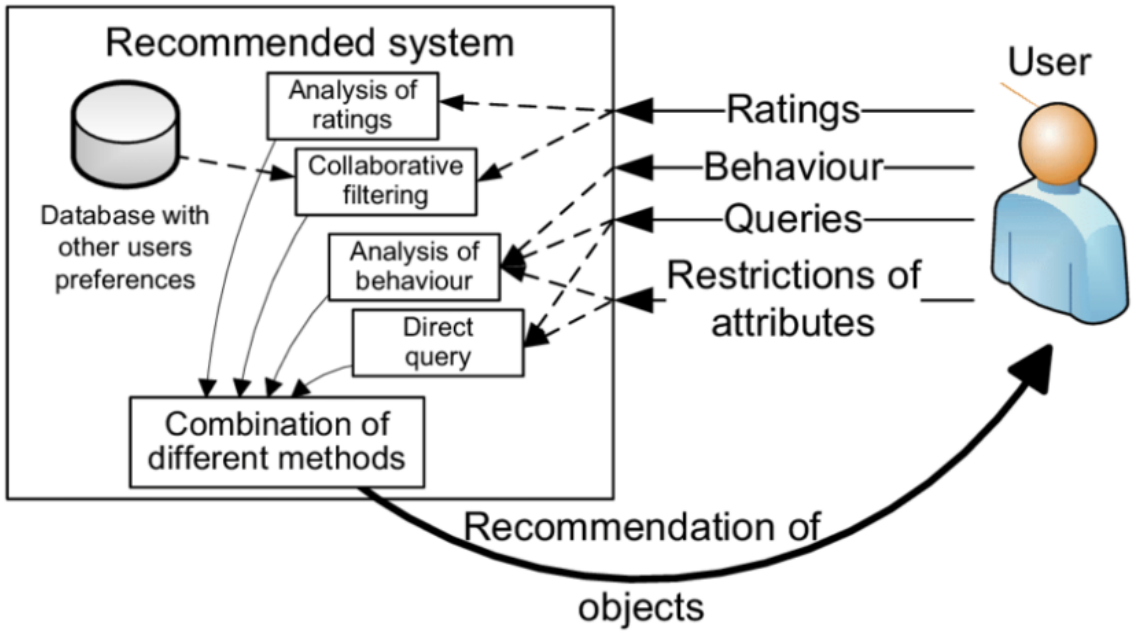

Figure 1.1 Recommender System

1.2 Collaborative Filtering

User-based collaborative filtering is a recommendation technique that focuses on identifying and recommending items to a target user by finding other users with similar preferences and behaviors. It operates under the assumption that users who have liked or interacted with similar items in the past will continue to have similar preferences in the future.

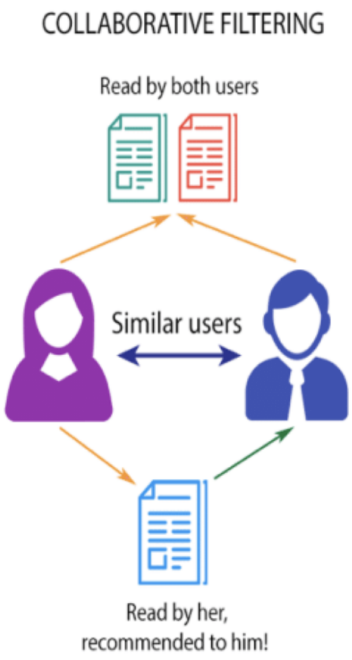

Figure 1.2.1 Collaborative Filtering

The first step is to create a user-item preference matrix. This matrix represents users in rows and items in columns, with the cells containing user ratings, interactions, or preferences for the corresponding items. It can be sparse because not all users have interacted with all items. To find similar users, the system calculates the similarity between the target user and other users. Various similarity metrics can be used for this purpose, with cosine similarity and Pearson correlation coefficient being common choices. These metrics measure how closely the preferences of two users align.

Once user similarities are calculated, the system identifies a set of the most similar users to the target user. This set is often referred to as the "neighborhood" of the target user. Recommendations are generated by looking at the items that the users in the neighborhood have liked or interacted with, but the target user has not. The system suggests these items to the target user, assuming that if similar users liked those items, the target user may also like them.

User-based collaborative filtering has its advantages, including the ability to provide personalized recommendations without needing detailed information about items. However, it also has some limitations and challenges, including the scalability issue when dealing with a large number of users and the "cold start" problem when new users join the system and don't have a sufficient history of interactions.

It's important to note that user-based collaborative filtering works well when users' preferences are stable over time and when there is enough data to identify meaningful user similarities. In practice, many recommender systems combine user-based collaborative filtering with other techniques, such as item-based collaborative filtering or content-based filtering, to improve the quality and robustness of recommendations.

1.3 Content-Based Filtering

Content-based filtering is a recommendation technique that focuses on making personalized recommendations to users by analyzing the characteristics or attributes of items and matching them to the user's profile of preferences. This approach relies on the idea that if a user has shown interest in certain items in the past, the system can recommend similar items based on shared features.

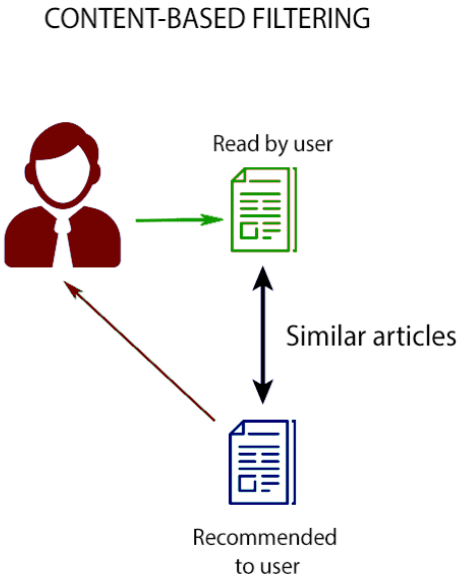

Figure 1.3.1 Content-Based Filtering

For each item in the system (e.g., movies, products, articles), relevant features or attributes are extracted. These features can vary depending on the type of items and may include information such as genre, actors, director, keywords, author, genre, product description, and more. This is known as Item Feature Extraction.

To create a user profile, the system looks at the items the user has previously interacted with or liked. It then assigns weights to these items based on the importance of the features in the user's past interactions. For example, if a user frequently watches action movies, the system might assign a higher weight to the "action" feature. The system generates recommendations by identifying items with features that match the user's profile. It scores and ranks items based on the similarity between their features and the user's preferences. Items with the highest similarity scores are recommended to the user.

Content-based filtering has several advantages, including the ability to provide recommendations even when there is limited user-item interaction data. It can also offer serendipitous recommendations by suggesting items with similar features that the user may not have discovered otherwise.

However, content-based filtering also has some limitations, such as the potential for overspecialization, where recommendations may be too similar to the user's past interactions, and it may not capture diverse user interests. To address these limitations, many recommender systems use hybrid approaches that combine content-based filtering with other techniques like collaborative filtering to provide a more well-rounded recommendation experience.

Content-based filtering is commonly used in various domains, including e-commerce, news websites, music streaming platforms, and more, where item features can be analyzed and used to make personalized recommendations to users.

1.4 Cold Start Problem

The "cold start problem" is a common challenge in recommender systems that refers to the difficulty of making accurate recommendations for new users or new items with limited historical interaction data. This problem arises when the system lacks sufficient information about the preferences and behavior of these new users or items.

The cold start problem can manifest in two main forms: cold start for new users and cold start for new items.

When a new user joins a recommender system, the system has little or no information about their preferences, as they haven't provided any ratings, reviews, or interaction history. Without historical data to analyze, it's challenging to make personalized recommendations for new users. The system may resort to making generic, non-personalized recommendations or suggestions based on demographic information (age, location, gender), which are often less accurate.

When new items are introduced to the system, they lack historical interaction data, and the system cannot assess their suitability for specific users. Since the system relies on past user interactions to understand and recommend items, it may not be able to suggest new items accurately, leading to potential underexposure for these items.

1.5 Hybrid Recommendation System

A hybrid recommendation system is a type of recommender system that combines multiple recommendation techniques or approaches to provide more accurate and diverse recommendations. By leveraging the strengths of different recommendation methods, hybrid systems aim to overcome the limitations and challenges associated with individual techniques. There are various ways to design hybrid recommendation systems, but the most common approaches include: Weighted Hybrid, Switching Hybrid, Feature Combination.

Hybrid recommendation systems have several advantages. They can provide more accurate and diverse recommendations, improve robustness, and handle various recommendation scenarios. For example, a hybrid system might perform well in situations where there is limited user-item interaction data, a cold-start problem (new users or items), or when the recommendation task involves diverse item categories.

Hybrid systems are commonly used in real-world applications, including e-commerce, content streaming platforms, and news websites. The choice of hybridization approach depends on the specific use case and the available data, and designing an effective hybrid

recommendation system often involves experimentation and testing to find the best combination of techniques for a given context.

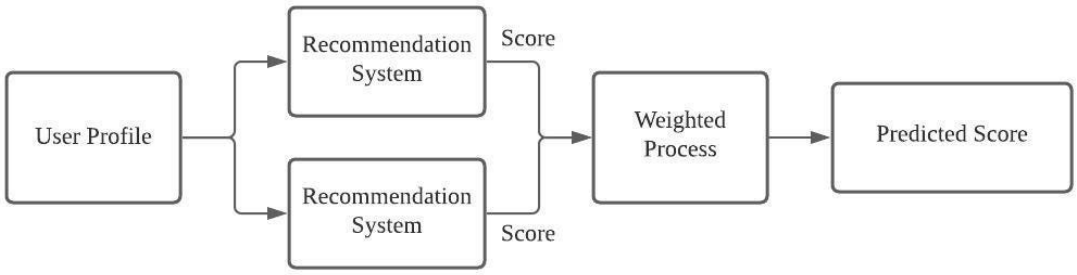

Figure 1.5.1 Hybrid Recommendation System

1.6 Illustration with an Example

1. Popularity Based-Recommender: Consider the ratings for an item given by five different users are (4, 5, 3, 4, 5). The mean rating needs to be calculated for the item.

Mean = (4 + 5 + 3 + 4 + 5) / 5 = 4.2. The average rating or mean rating of that item in 4.2.

2. Singular Value Decomposition: Finding the SVD of A, $U\Sigma V^T$,

where A = 3 2 2

2 3 −2

First we compute the singular values σi by finding the eigenvalues of AAT . AAT = 17 8 8 17. The characteristic polynomial is

$\det(AAT - \lambda I) = \lambda 2 - 34\lambda + 225 = (\lambda - 25)(\lambda - 9)$,

so the singular values are $\sigma 1 = \sqrt{25} = 5$ and $\sigma 2 = \sqrt{9} = 3$.

Now we find the right singular vectors (the columns of V) by finding an orthonormal set of eigenvectors of ATA. It is also possible to proceed by finding the left singular vectors (columns of U) instead.

The eigenvalues of ATA are 25, 9, and 0, and since A^TA is symmetric we know that the eigenvectors will be orthogonal.

For $\lambda = 25$, we have $A^TA - 25I = \begin{matrix} -12 & 12 & 2 \\ 12 & -12 & -2 \\ 2 & -2 & -17 \end{matrix}$

which row-reduces to $\begin{matrix} 1 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$

A unit-length vector in the kernel of that matrix is $v1 = \begin{matrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{matrix}$

For $\lambda = 9$ we have $A^TA - 9I = \begin{matrix} 4 & 12 & 2 \\ 12 & 4 & -2 \\ 2 & -2 & -1 \end{matrix}$

which row-reduces to $\begin{matrix} 1 & 0 & -1/4 \\ 0 & 1 & 1/4 \\ 0 & 0 & 0 \end{matrix}$

A unit-length vector in the kernel is $v2 = \begin{matrix} 1/\sqrt{18} \\ -1/\sqrt{18} \\ 4/\sqrt{18} \end{matrix}$

For the last eigenvector, we could compute the kernel of A^TA or find a unit vector perpendicular to v1 and v2. To be perpendicular to $v1 = \begin{matrix} a & b & c \end{matrix}$ we need $-a = b$.

Then the condition that $v2^T\ v3 = 0$ becomes $2a/\sqrt{18} + 4c/\sqrt{18} = 0$ or $-a = 2c$.

So $v3 = \begin{matrix} a & -a & -a/2 \end{matrix}$ and for it to be unit-length we need $a = 2/3$ so $v3 = \begin{matrix} 2/3 & -2/3 & -1/3 \end{matrix}$.

So at this point we know that $A = U\Sigma V^T =$

$$U \;.\; \begin{matrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{matrix} \;.\; \begin{matrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{matrix}$$

Finally, we can compute U by the formula $\sigma ui = Avi$, or $ui = 1\ \sigma Avi$.

This gives $U = \begin{matrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{matrix}$

So SVD is: $A = U\Sigma V^T =$

$$\begin{matrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{matrix} \cdot \begin{matrix} 5 & 0 & 0 \\ 0 & 3 & 0 \end{matrix} \cdot \begin{matrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{18} & -1/\sqrt{18} & 4/\sqrt{18} \\ 2/3 & -2/3 & -1/3 \end{matrix}$$

3. Tf-Idf:

Imagine the term t appears 20 times in a document that contains a total of 100 words. Term Frequency (TF) of t can be calculated as follows:

TF = 20 / 100 = 0.2

Assume a collection of related documents contains 10,000 documents. If 100 documents out of 10,000 documents contain the term t, Inverse Document Frequency (IDF) of t can be calculated as follows

IDF = log(10000/100) = 2

Using these two quantities, we can calculate the TF-IDF score of the term t for the document

TF-IDF = TF * IDF

= 0.2 * 2

= 0.4

2. LITERATURE REVIEW

Aggarwal, et al[1] did research on Collaborative Recommendation Systems. While the approaches capture the change in user interests over time and are direct and intuitive to implement, users with few ratings cause data sparsity which makes the model prone to skewed correlations. This is also known as “Grey sheep issue”. Probabilistic predictions method(Nearest Neighbors method) and Decision Trees are reviewed and both of the methods do not perform well with sparse data. **M Viswa Murali, et al[2]** developed a similar system but it too failed to address the cold start problem.

Abinash Pujahari, et al[4] proposed various methods for Content Based Recommender Systems including Decision Trees, Bayesian Classifiers and Relevance feedback. Although the methods have proven to provide adequate results in a specific context, they lack an important factor needed in recommender systems: adaptability in new input data. **Yifan Tai, et al[3]** proposed a new method for Content Based Filtering Techniques in Recommendation System using user preferences. This method has limited ability to expand on existing user interests.

Keshetti Sreekala[5] project explores the Popularity Based Recommendation method. This approach helps recommend items that are trending and have a high average rating. Although this method promotes the items that are bought by a lot of people and have good ratings, it fails to consider user preferences. Although, this method performs when there is less user data and it is adaptable to adding new users and items. The algorithm used is a non-personalized recommendation algorithm that considers user interest that is present in the dataset. The Mean algorithm is also referred to as the item average algorithm or the POP algorithm. Whenever the prediction is needed for an item or product it is calculated based on the average of polls of all users of that item or product.

Shaanya Singh, et al[6] developed a project named “RecNN” which is a Deep Neural Network based Recommendation System. Using the project, they extended the applications of Neural Networks to Recommender Systems. In this project, they evaluated the performance of a revised version of a deep learning-based recommender system for movies and books using multiple fully connected dense layered neural net embeddings-based structures that primarily ensembles deep neural networks integrated

alongside embedding layers and their dot product values. The results obtained show that their approach is a promising solution when compared with the independent memory-based collaborative filtering methods or content-based methods. But Neural Networks are heavily computationally and memory intensive.

Mustansar ali Ghazanfar, et al[7] developed a scalable, accurate hybrid recommender system. This project uses the Hybrid approach for the recommendations to obtain more accurate results. This method switches between recommendation algorithms based on context. It uses collaborative filtering for existing users and content-based filtering for new users. Although it is more scalable than collaborative filtering or content-based filtering, it has the cold-start problem.

Gediminas Adomavicius, et al[8] conducted research on various recommendation approaches and reviewed the limitations and benefits of each approach. While mentioning the possible extensions to the existing algorithms, they proposed a Hybrid approach that combines three approaches namely: Popularity based approach, Collaborative Filtering and Content-Based Filtering. The results from the three approaches are combined using weighted-average technique. This method so far performs the best against the individual algorithms.

2.1 Comparison of existing methods

Sl. no	Author(s)	Method	Advantages	Disadvantages
1	J. Ben Schafer, Dan Frankowski, Jon Herlocker, Shilad Sen	Collaborative Filtering Recommender Systems	Captures the change in user interests over time, direct and intuitive to implement	Users with few ratings cause data sparsity which makes the model prone to skewed correlations (Grey sheep issue)
2	R. Manjula, A. Chilambuchelvan	Content Based Filtering using user preferences	Model does not need data about users	Limited ability to expand on existing user interests
3	Keshetti Sreekala	Popularity based recommendation	Immune to cold start problem	Recommendations are not personalized
4	Shaanya Singh, Maithili Lohakare, Keval Sayar, Shivi Sharma	Neural Networks	Multitask learning (train model on multiple objectives)	Computationally intensive
5	Mustansar ali Ghazanfar, A. Prugel-Bennett	Switching Hybrid Recommendation	Combines the advantages of both content based and collaborative filtering techniques	Cold start problem

3. Hybrid Recommendation System

The Hybrid Recommendation System is a combination of Popularity Based Recommender, Collaborative Filtering Recommender, and Content-Based Recommender. The results are calculated separately using the three approaches and the recommendations are combined using weighted-average technique. This approach combines the advantages of the three approaches and also overcomes the cold-start problem that is prevalent in Recommender Systems.

3.1 Popularity Based Recommendations

Popularity model is a fundamental model based on the popularity and rating of an item given to it by the user. It is a non-personalized algorithm. The Popularity model recommends items to the user according to the current trends. It may suggest to the user those items which are the most common or the most selling items. The main drawback to this model is the lack of personalization. Even though we know the behavior of the user, we cannot recommend him accordingly. This model will make the same item recommendations to all the users regardless of their history. Simple popularity model works best for new users or those users using a new account. Since their item history will be null, this model will suggest them those items that are popular in general. The user has a higher chance of liking the item since it has a high rating and is liked in general. Each user will be recommended with the items according to the current trend.

The ratings of the users are taken and the mean is calculated for the ratings and they are sorted according to the averages.

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N}$$

Figure 3.1.1 Mean Formula

3.2 Collaborative Filtering

Collaborative filtering is a popular technique used in recommender systems to make recommendations by leveraging the preferences and behaviors of users. It is based on the idea that users who have interacted with or liked similar items in the past will continue to have similar preferences in the future.

The technique used is Singular Value Decomposition (SVD). Singular Value Decomposition is a mathematical technique used in linear algebra and data analysis. It is widely used in various fields, including recommendation systems, image processing, and dimensionality reduction. In the context of recommendation systems, SVD is often used to perform collaborative filtering, a technique for making personalized recommendations based on user-item interaction data.

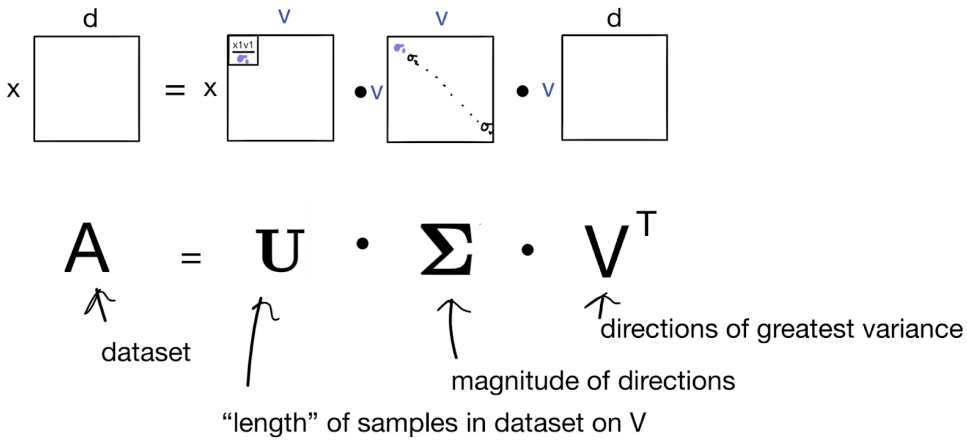

Figure 3.2.1 SVD Formula

A matrix is constructed with users as rows and ratings as columns. The matrix is decomposed into three matrices and it is recomposed by performing dot product between the matrices. The decomposed matrix fills in the missing data and new ratings are obtained based on similarity. The ratings are then sorted and the top recommendations are made to the user.

3.3 Content-Based Filtering

Content-based filtering is a recommendation system technique that suggests items to users based on the attributes and features of the items themselves. It is a type of recommendation system that relies on analyzing the content or characteristics of items, such as movies, products, or articles, and matching these features to a user's profile to make personalized recommendations.

The technique used is Term Frequency-Inverse Document Frequency (TF-IDF). It is a numerical statistic used in information retrieval and text mining to evaluate the importance of a term within a document relative to a collection of documents. It is often employed in natural language processing, text classification, and search engines. Its applications can also be extended to Recommender Systems. The technique is a combination of 2 parts: Term Frequency and Inverse Document Frequency.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Figure 3.3.1 TF-IDF formula

3.3.2 Term Frequency

Term frequency measures how frequently a term appears in a document. It is calculated as the number of times a term (word) occurs in a document divided by the total number of terms in that document. The idea is to give higher weight to terms that appear more frequently within a document, assuming that they are more relevant.

$$TF = \frac{Number\ of\ times\ a\ word\ "X"\ appears\ in\ a\ Document}{Number\ of\ words\ present\ in\ a\ Document}$$

Figure 3.3.2.1 Term Frequency

3.3.3 Inverse Document Frequency

Inverse document frequency assesses the importance of a term within a collection of documents. It is computed as the logarithm of the total number of documents divided by the number of documents containing the term.The intention is to give higher weight to terms that are relatively rare in the entire document collection, as they are more discriminative.

$$IDF = log\left(\frac{Number\ of\ Documents\ present\ in\ a\ Corpus}{Number\ of\ Documents\ where\ word\ "X"\ has\ appeared}\right)$$

Figure 3.3.3.1 Inverse Document Frequency

3.3.4 TF-IDF

TF-IDF is commonly used in information retrieval to rank documents for a given query. Documents with higher TF-IDF scores for query terms are considered more relevant. It is used in text classification tasks to represent documents as feature vectors, where each term's TF-IDF score becomes a feature.

It is obtained by multiplying tf and idf score. **TF-IDF = TF * IDF**

3.3.5 Cosine Similarity

Cosine similarity is a metric used to measure the cosine of the angle between two non-zero vectors in an n-dimensional space. In the context of natural language processing and recommendation systems, cosine similarity is often used to compare the similarity between two vectors representing textual documents or user-item interactions.

To use cosine similarity, you first represent the entities you want to compare as vectors. In the context of text data, documents, sentences, or words are often represented as vectors in a high-dimensional space, with each dimension corresponding to a different term or feature. Each vector typically represents the term frequencies of the words in the documents. This means that each dimension of the vector corresponds to a word in the vocabulary, and the value in that dimension is the frequency of the word in the document. These vectors are often referred to as term frequency vectors.

To calculate the cosine similarity between two vectors, first calculate the dot product of the two vectors, which is the sum of the products of their corresponding elements. Calculate the magnitude (length) of each vector, which is the square root of the sum of the squares of its elements. Divide the dot product by the product of the magnitudes of the two vectors. The result is a value between -1 and 1, where 1 indicates that the vectors are identical, 0 means they are orthogonal (unrelated), and -1 implies they are diametrically opposed.

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Figure 3.3.5.1 Cosine Similarity

Cosine similarity values range from -1 (completely dissimilar) to 1 (completely similar). A value of 0 implies that the vectors are orthogonal or have no apparent similarity. Larger positive values indicate greater similarity, while larger negative values indicate greater dissimilarity.

3.4 Weighted Average

The results from the individual approaches are obtained and they are combined using weighted average. Weighted average is calculated as follows:
Weighted Average = (Weight1 * Value1 + Weight2 * Value2 + ... + WeightN * ValueN) / (Weight + Weight + ... + WeightN)

The weights can be adjusted according to how the recommender system is supposed to act. Assigning more weight would give more preference to an approach and vice-versa. The averages are sorted and the top recommendations are made to the user.

4. IMPLEMENTATION

4.1 Packages Used

1. **numpy:** NumPy (Numerical Python) is a fundamental Python library for numerical computing
2. **pandas:** Pandas can be used to read CSV data and store them in dataframes and perform operations on the dataframe
3. **surprise:** The Surprise library in Python, also known as scikit-surprise, is a popular and easy-to-use library for building and evaluating recommendation systems
4. **sklearn.feature_extraction.text:** The package consists of the implementation of Tf-Idf
5. **sklearn.metrics.pairwise:** The package consists of the implementation of cosine similarity which is used to calculate the similarity between items

4.2 Attributes

1. **generate_id():** Converts item ids to unique integer ids
2. **rmse():** Calculates the Root Mean Square Error
3. **mae():** Calculates the Mean Absolute Error

4.3 Variables

1. **df:** Dataset containing users and ratings
2. **item_ratings:** Subset of dataset containing only user id's and ratings
3. **pbr:** Popularity Based recommendations
4. **cb:** Collaborative Filtering recommendations
5. **cbf:** Content-Based Filtering recommendations
6. **unique_titles:** Subset of dataset containing only product titles
7. **score_series:** Matrix of cosine similarities
8. **hybrid_rec:** Combined ratings of three approaches
9. **predicted:** New Hybrid recommendations
10. **rmse_hybrid:** Root Mean Square Error of Hybrid System

11. **mae_hybrid:** Mean Absolute Error of Hybrid System

4.4 Dataset

The dataset consists of ratings of products that are bought on the website Amazon. This data is scraped from the web and the dataset is obtained from kaggle.
https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset

4.4.1 Dataset Attributes

1. **item_id:** Unique item identifier for each product
2. **title:** The title of the product displayed on the website
3. **user_id:** Unique identifier for each user
4. **rating:** Numerical rating value from a scale of 0 to 5
5. **main_cat:** Main category of the product
6. **sub_cat:** Subcategories of the product

5. EXPERIMENT RESULTS

5.1 Experimentation Screenshots

```
pbr = item_ratings.loc[item_ratings['item_id'] == item, 'rating'].values[0]
```

```
i = np.where(predictions['item_id'].values == item)[0][0]
cfr = predictions.at[predictions.index[i],'rating']
```

```
cbf = score_series.loc[score_series['item_id'] == item, 'rating'].values[0]
```

```
predicted = pbr * 0.2 + cfr * 0.4 + cbf * 0.4
```

```
import math
def rmse(prediction, actual):
    return math.sqrt((prediction-actual)**2)

def mae(prediction, actual):
    return abs(prediction-pbr)
rmse_hybrid = round(rmse(predicted, actual), 4)
mae_hybrid = round(mae(predicted, actual), 4)
print(f"rmse: {rmse_hybrid}")
print(f"mae: {mae_hybrid}")
```

```
rmse: 0.4297
mae: 0.2297
```

```
hybrid_rec.sort_values(by='r', ascending=False)['item_id'].head()
```

```
7803     3630128198
4052     1867199229
12378    5800311307
10822    5047884572
18903    8952690798
Name: item_id, dtype: int64
```

5.2 Parameters

The parameters considered for evaluating the proposed method are Root Mean Square Error and Mean Absolute Error. They are two common metrics used to evaluate the performance of prediction or regression models, including those used in recommendation systems. These metrics assess the accuracy of predicted values compared to actual values in a dataset.

5.2.1 Root Mean Square Error

RMSE measures the square root of the average of the squared differences between predicted and actual values. It is a popular metric for assessing the overall model accuracy, and it emphasizes larger errors more than smaller ones. A lower RMSE indicates a more accurate model. It is more sensitive to outliers than MAE, as it squares the error terms.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

$\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n$ are predicted values

$y_1, y_2, \ldots, y_n$ are observed values

n is the number of observations

Figure 5.2.1.1 RMSE Formula

5.2.2 Mean Absolute Error

MAE measures the average of the absolute differences between predicted and actual values. It provides a simple and straightforward measure of prediction accuracy and is less sensitive to outliers than RMSE. A lower MAE indicates a more accurate model. It treats all errors equally and is a good choice when extreme errors should not be given extra weight.

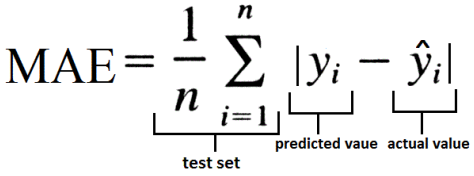

Figure 5.2.2.1 MAE Formula

5.2.3 Comparison

RMSE tends to be higher than MAE because it emphasizes larger errors more. It is useful when larger errors should be penalized more.

MAE treats all errors equally and is more robust to outliers. It provides a better understanding of the average magnitude of prediction errors.

In the context of recommendation systems, both RMSE and MAE can be used to evaluate the quality of recommendations.

6. DISCUSSION OF RESULTS

Analysis and comparison were conducted between Collaborative filtering, Content-Based Filtering and the Hybrid Recommendation System. The experimentation utilized 3.9 GB storage and 3.8 GB RAM. The implementation was carried out in the Kaggle code environment.

6.1 Actual Ratings vs Predicted Ratings

The figure 6.1.1 shows the comparison of actual ratings and the predicted ratings of 10 users.

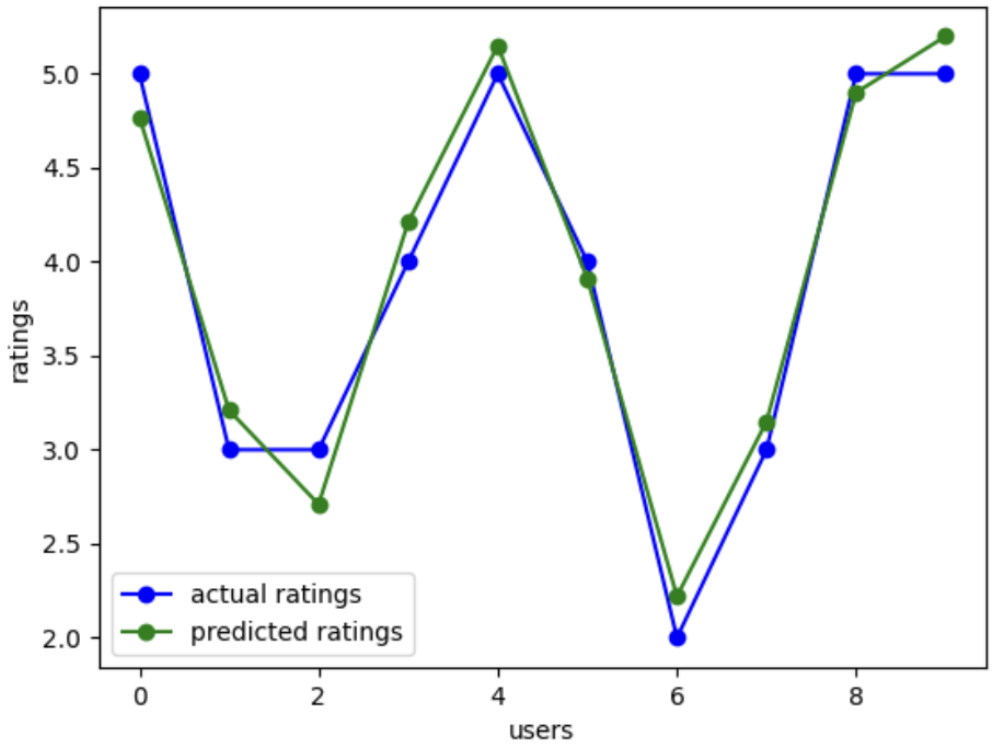

Figure 6.1.1 Actual ratings vs Predicted ratings

6.2 Accuracy Comparison

The table shows the comparison between RMSE and MAE scores of the existing method and the proposed Hybrid method

Table 6.2.1 Comparison of Accuracy Scores

Parameter	Mathematical function	Existing value	Improved value
Root Mean Square Error	RMSE	1.4081	0.4279
Mean Absolute Error	MAE	1.1520	0.2279

Figure 6.2.2 and Figure 6.2.3 show the comparison of accuracy scores for 10 different users whose interaction is low

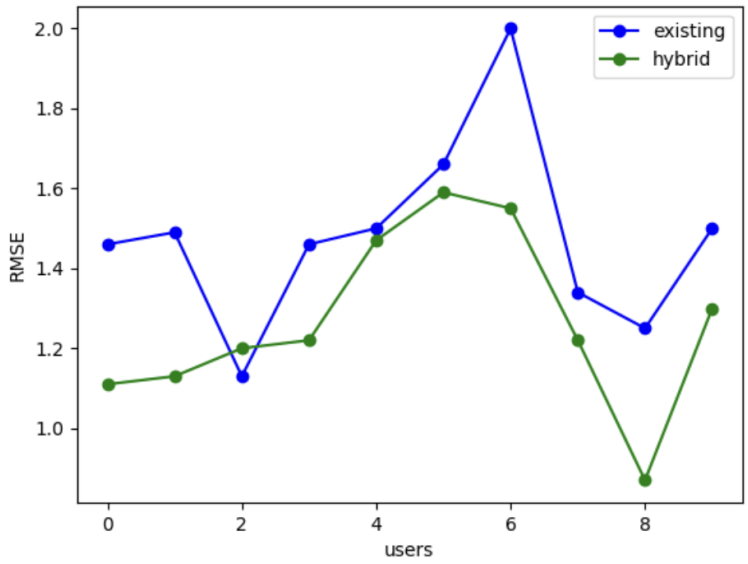

Figure 6.2.2 Comparison of RMSE

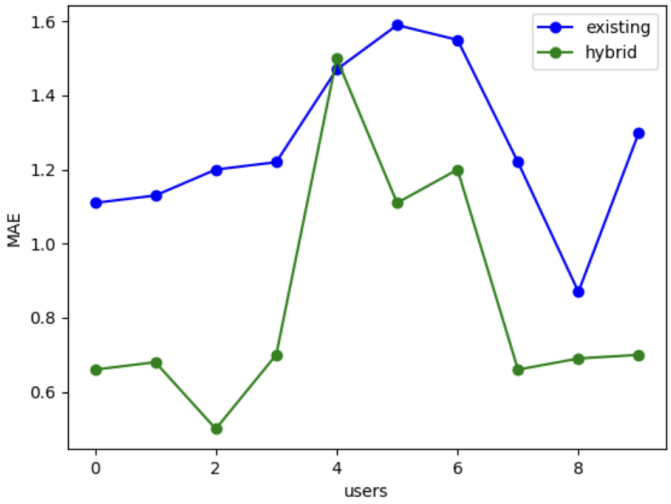

Figure 6.2.3 Comparison of MAE

7. CONCLUSION

In conclusion, our proposed Hybrid Recommendation System, combining Popularity Based Recommendations, Collaborative Filtering and Content-Based Filtering, presents a robust and efficient solution for the “cold-start problem”.

The Singular Value Decomposition makes the algorithm work better with sparse data. Combining the results with Content-Based recommendations makes more accurate recommendations when the user database is less and there is less information about users to make similarities between the users. The Popularity recommendations pushes the trending items to the top of the recommendations list.

Compared to the existing recommendation methods, the Hybrid Recommendation System performs better in terms of accuracy. When the accuracy scores of the approaches are compared, it is clear that the Hybrid System gets a lower error score than the existing methods. When the results of a user are taken, the RMSE and MAE scores are 1.40 and 1.52 respectively while the scores of the Hybrid system are 0.42 and 0.22 respectively. A lower score means that the system is more accurate.

These results show that the model performs better than the existing methods when less user data is present and it can be shown that the Hybrid system addresses the cold-start problem better than the existing methods.

8. REFERENCES

[1] Aggarwal, C.C., Wolf, J., Wu, K.L., Yu, P.S. (1999) “A New Graph-Theoretic Approach to Collaborative Filtering”, Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining, San Diego, California, pp. 201–212. ACM, New York

[2] Murali, M. V., Vishnu, T. G., & Victor, N. (2019) A Collaborative Filtering based Recommender System for Suggesting New Trends in Any Domain of Research. 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)

[3] Y. Tai, Z. Sun, and Z. Yao (2021) “Content-Based Recommendation Using Machine Learning,” International Workshop on Machine Learning for Signal Processing

[4] A. Pujahari and V. Padmanabhan (2014) "An Approach to Content Based Recommender Systems Using Decision List Based Classification with k-DNF Rule Set," 2014 International Conference on Information Technology

[5] Keshetti Sreekala (2020) “Popularity Based Recommendation System”, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online), Volume-9 Issue-3

[6] S. Singh, M. Lohakare, K. Sayar and S. Sharma (2021) "RecNN: A Deep Neural Network based Recommendation System", 2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)

[7] Ghazanfar, Mustansar ali & Prugel-Bennett, A.. (2010). A Scalable, Accurate Hybrid Recommender System. 3rd International Conference on Knowledge Discovery and Data Mining, WKDD 2010

[8] Adomavicius, G., Tuzhilin, A. (2005) Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749