ITEM RECOMMENDATION USING BICLUSTER APPROACH

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology
in
Computer Science and Engineering/Information Technology

By

SHIVAM SINGH NEGI (191260)

Under the supervision of

Dr. Rakesh Kanji
to

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology
Waknaghat, Solan-173234, Himachal Pradesh

Candidate's Declaration

I hereby declare that the work presented in this report entitled " ITEM RECOMMENDATION USING BICLUSTER APPROACH" in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering/Information Technology submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from July 2022 to May 2023 under the supervision of Dr. Rakesh Kanji (Assistant Professor (SG) and Computer Science and Engineering/Information Technology).
The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)
SHIVAM SINGH NEGI, 191260.

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)
Supervisor Name -Dr. Rakesh Kanji
Designation -Assistant Professor (SG)
Department name –Computer Science and Engineering/Information Technology
Dated: 01/MAY/2023

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: ________________________ __**Department:** _______________ **Enrolment No** ________

Contact No. ___________________________**E-mail.** _________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at ....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

ACKNOWLEDGEMENT

I would like to thank and express our gratitude to our Project supervisor Dr. Rakesh Kanji for the opportunity that he provided us with this wonderful project "Item Recommendation Using Bicluster Approach". The outcome would not be possible without his guidance. This project taught me many new things and helped to strengthen concepts of Machine Learning. Next, I would like to express my special thanks to the Lab Assistant for cordially contacting us and helping us in finishing this project within the specified time. Lastly, I would like to thank my friends and parents for their help and support.

Shivam Singh Negi (191260)

Table of Content

List of Abbreviations

Sr. No.	Abbreviations	Full Form
1	TP	True Positive
2	FP	False Positive
3	FN	False Negative
4	CoClust	Co-clustering
5	MAE	Mean Absolute Error
6	RMSE	Root Mean Square Error
7	DBSCAN	Density-based spatial clustering of applications with noise.
8	BiMax	Binary inclusion-Maximal
9	NBCF	Nearest- bicluster collaboratie filtering

List Of Figures

ABSTRACT

By using the complex, highly dimensional data analysis technique known as biclustering, rows and columns are simultaneously grouped. In the context of recommendation systems, biclustering may be used to identify subgroups of users and items that have strong linkages and are thus likely to make suitable recommendations for one another. This method has a number of advantages over traditional recommendation systems, including the ability to handle noisy and sparse data and the possibility to uncover previously unknown patterns of user-item interactions. The essential concepts behind biclustering-based recommendation systems are outlined in this abstract, along with some recent advancements. We also discuss some of this strategy's disadvantages and benefits and provide suggestions for more research. We believe that biclustering-based systems for recommendation might significantly improve the accuracy and significance of item suggestions in a range of diverse applications.

CHAPTER-1

INTRODUCTION

1.1 Introduction

Recommendation systems are computer algorithms that analyze data and provide personalized recommendations to users. These systems are designed to predict preferences of a user based on previous actions, for instance their surfing history, purchase history, ratings, and reviews.

Recommendation systems are used in a variety of contexts, such as e-commerce, streaming services, and social media and advertising. For example, e-commerce websites like Amazon and eBay use recommendation systems to suggest products that a user may be interested in based on their purchase history and browsing behavior. Recommendation algorithms are used by streaming services like Netflix and Spotify to make movie and TV programme suggestions, and songs that a user may like based on their viewing and listening history.

The importance of recommendation systems in the modern world cannot be overstated. With the explosion of data, it has become increasingly difficult for users to find relevant and personalized content. Recommendation systems help users to discover new content that they may be interested in, which can improve user engagement and satisfaction. Additionally, recommendation systems can help businesses to increase revenue by promoting relevant products and services to users.

In summary, recommendation systems are a powerful tool for personalizing user experiences and increasing engagement and revenue for businesses. As data continues to grow, the importance of recommendation systems will only continue to increase in the modern world.

1.2 Problem Statement

Applications for mobile devices, display advertising, online gaming, and e-commerce have all widely used recommendation systems. Although significant progress has been made recently, studies into creating effective algorithms to improve recommender system efficiency is still ongoing. One significant use is ranking a list of items that others may access. It is more likely that a customer will decide to click on and purchase anything from a recommended list based on how relevant it is. This increases business income and has broad commercial ramifications. The top-n ranking issue is used to represent the research problem, and the number of hits in the top-n ranked list is used to measure performance. One approach to allow the system to rate the objects is through collaborative filtering (CF) technology. Although many solutions to this issue have been put forth, they are all hampered by either issues with computation speed and scalability or the necessity of model tuning (especially for model-based approaches). In this paper, we provide a neighborhood-based bicluster technique for generating top-n suggestions. Investigations are conducted on both standard item similarity and bicluster similarity.In a dense submatrix wherein each user engages with each object, a bicluster is a subset of users and objects that collectively make up the submatrix.

1.3 Objectives

The Item Recommendation system offers a mechanism for categorising users with similar interests and searching for content that would be of great interest to distinct groups of users, before creating various types of lists and making interesting recommendations to the individual based on the content they enjoy. The recommender system's major goal is to apply techniques that offer a biclustering method to determine the set of films that each user enjoys for a certain group of users. The movies that have a high probability of being liked by an overall group of users will be suggested to the user by the recommender at the end, and then we will try to find the users with distinct interests using the data collected through different activities, and using collaborative filtering, we will evaluate all those users who have the same type of interests to arrive at the final set of movies to be recommended to the users individually.So, we will use various kinds of recommender filtering approaches and then compare and contrast the results obtained by different methods, attempting to improve the results as the dataset for the set of movies grows progressively larger above the computational bound of the system, which is generally a drawback on large datasets.

1.4 Methodology

There are various methods to build a recommender system. They are as follows:

1) Demographic Filtering:

Demographic filtering is a technique used in recommendation systems that considers the demographic information of the users to make recommendations. This information can include age, gender, location, income, education, and other factors. Demographic filtering assumes that people with similar demographics may have similar interests and preferences, so it recommends items that are popular among users with similar demographics.

Demographic filtering can be useful for making recommendations in contexts where people's interests are closely tied to their demographics, such as in movies or music genres that appeal to certain age groups or genders. However, demographic information alone may not be sufficient to provide accurate recommendations. People within the same demographic group can have very different interests and preferences, and demographic information may not capture these individual differences. As a result, it is frequently used in conjunction with various other recommendation approaches, like collaborative filtering or content-based filtering, to produce recommendations that are more accurate. Overall, demographic filtering is a useful approach for making recommendations in certain contexts, but it should be used in conjunction with other methods to provide personalized recommendations that reflect individual preferences.

Algorithm for demographic filtering .

Input:

- User profile data (e.g., age, gender, location)

- All items in the system, each with demographic attributes (e.g., age range, gender, location)

Output:

- Filtered items that match user's demographic criteria

Algorithm:

1. Initialize an empty list called "filtered_items".

2. For each item in the system, do the following:

- Check if the item matches the user's age range.
- Check if the item is for the user's gender or is gender-neutral.
- Check if the item is available in the user's location.

If the item satisfies all of the above conditions, add it to the "filtered_items" list.

3.Sort the "recommended_items" list by some relevance metric (e.g., popularity, ratings, or relevance to user's past behavior).

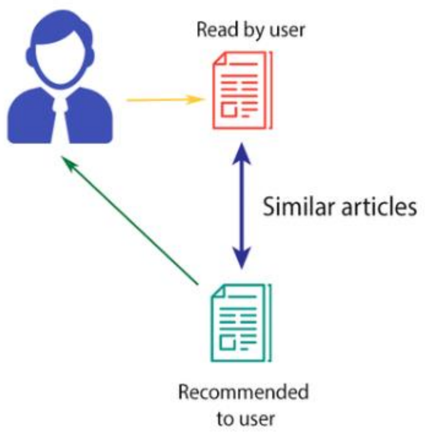
4. Return the "filtered_items" list as the output.

This algorithm assumes that the items in the system have attributes such as age range, gender, and location, and that the user has provided their own demographic information. The algorithm iterates over all items and filters them based on whether they match the user's demographic criteria. Finally, the filtered items are returned as the output.

2) Content Based Filtering System:

Content-based filtering is a recommendation system approach that leverages the attributes of the things being recommended to identify more comparable items. The theory behind content-based filtering is that if a consumer like one thing, they are likely to appreciate other goods with comparable attributes.

In a content-based filtering system, each item is represented by a set of features, such as genre, actor, director, or author. These features can be extracted from various sources, such as user ratings, item descriptions, or metadata. Once the features are identified, the system can then use a similarity measure, such as cosine similarity or Euclidean distance, to compare the features of the items and identify similar items. The content-based filtering method suggests products that are comparable to past favourites of the user. For example, if a user has already enjoyed a romantic comedy movie starring Jennifer Aniston, the algorithm might suggest further romantic comedies starring Jennifer Aniston or other performers with comparable attributes.



.

Fig.1.1-Content Based Filtering Method from [10]

So below mention is the algorithm for a Content-based filtering recommendation system based on a user movie rating matrix:

- Collect user data and item data.
- Create item profiles based on their features or attributes.
- Compute a user profile based on their past behavior and preferences.
- Compute item similarity scores based on their profiles.
- Generate recommendations by selecting the top items similar to the ones the user has interacted with or rated.
- Evaluate and refine the system using metrics such as precision, recall, or mean average precision.

A similarity score is a numerical value that quantifies the degree of similarity or dissimilarity between two items, based on their attributes or features. There are several similarity scores that can be used in Content-based filtering recommendation systems like:

1. **Cosine Similarity:**

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

Fig.1.2-Formula for cosine similarity from [14]

where A and B are two vectors representing the attributes of two items.

The cosine similarity is a result of the angle between two vectors where the vectors are non-zero, and it is defined in the inner product space as the dot product of the two vectors divided by the product of the Euclidean magnitude. Cosine similarity is commonly utilised to obtain preferable recommendations for users. It computes the cosine of the angle formed by two vectors and ranges from -1 to 1 (opposite to identical).

2. **Euclidean distance:**

It determines the n-dimensional space's scalar product of the straight-line separation between any two locations. Euclidean distance between two points (x1, y1) and (x2, y2) is calculated using the following formula:

Distance = sqrt((x2 - x1)^2 + (y2 - y1)^2)

3. **Jaccard similarity:**

It computes the similarity between two sets based on the size of their intersection over the size of their union. The formula for Jaccard similarity between two sets, A and B is:

$J(A,B) = |A \cap B| / |A \cup B|$

4. **Hamming distance:**

It computes the number of positions at which two strings of equal length differ. The formula for Hamming distance between two strings, A and B is:

$dH(A,B) = \#\text{ of positions where } A[i] \neq B[i]$

Advantages of Content-Based Filtering in Recommendation Systems:

1. Users may receive personalised suggestions through content-based filtering that reflect their unique tastes and hobbies.

2. It does not rely on the opinions or ratings of other users, which can be biased or unreliable.

3. Content-based filtering can recommend items that are new or not yet popular, which can help to promote diversity in the recommendation system.

4. It can be implemented using simple algorithms and does not require a huge amount of data for training the system.

Disadvantages of Content-Based Filtering in Recommendation Systems:

1. It can only propose goods that are similar to those that the user previously liked or rated, which can lead to a lack of variety among recommendations.

2. Content-based filtering is unable to suggest products that fall outside of the user's known tastes, which may reduce the likelihood of a chance discovery.

3. It can be difficult to capture the full range of user preferences and interests using only content-based features, which can limit the accuracy of the recommendations.

4. Content-based filtering may not work well for new or obscure items that do not have a lot of content-based data available.

5. It can be susceptible to the problem of overfitting, where the recommendation system becomes too narrowly focused on the specific features used in the content-based filtering algorithm.

3) Collaborative Filtering:

In recommendation systems, a technique called collaborative filtering is used to forecast a user's interests and preferences based on similarities in their behaviour and the behaviour of other users who have those interests. In other words, collaborative filtering uses data about past user behavior to identify patterns and similarities among users, and then uses this information to suggest new items or content that the user might like.

The importance of collaborative filtering in recommendation systems lies in its ability to provide personalized recommendations to users. By analyzing the preferences of similar users, collaborative filtering can identify items or content that a user is likely to enjoy, even if they have never interacted with that item before. This can lead to increased user engagement, as well as increased revenue for businesses that rely on recommendations to drive sales.

One key benefit of collaborative filtering is its ability to deal with the "cold start" problem, which occurs when a new user joins a system and has little or no data available for the system to use in making recommendations. By using the behavior of other similar users, collaborative filtering can still provide useful recommendations to new users, even when there is little or no data available about their preferences.

Overall, collaborative filtering is an important technique in recommendation systems because it helps to provide personalized recommendations to users, even in cases where there is limited data available. This can lead to increased user engagement and revenue for businesses, making it an important tool for many organizations.

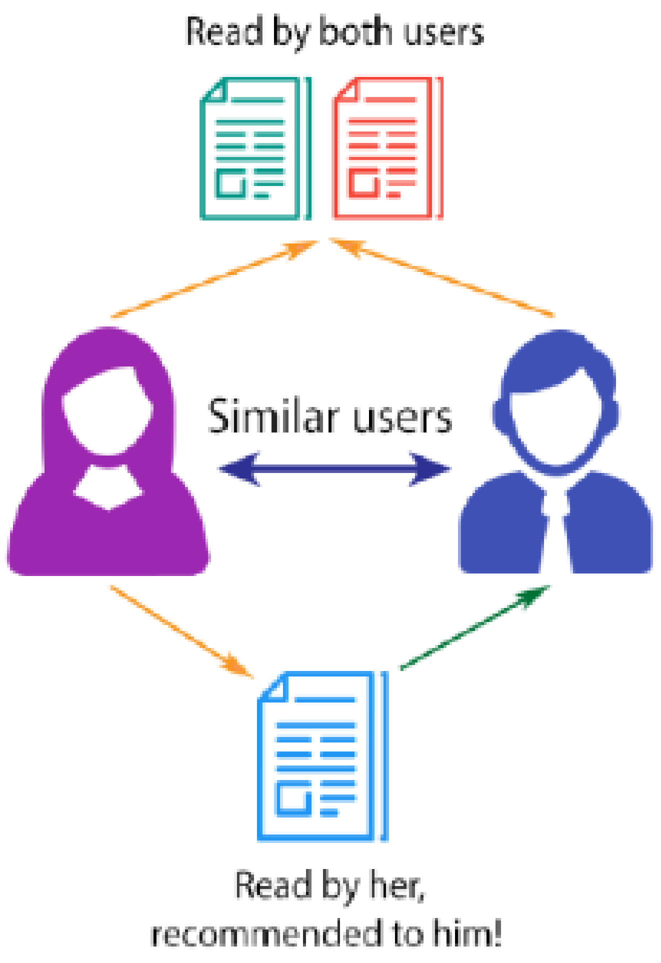


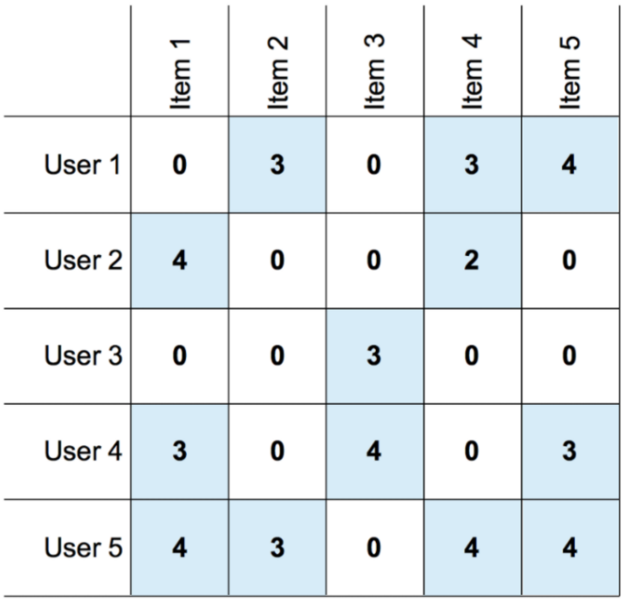
Fig.1.3- Illustrative of Collaborative Filtering from [10]

There are two main types of collaborative filtering methods:

1. **User-Based Collaborative Filtering:**

   In this method, similarities between users are calculated based on their past behavior, such as ratings or reviews. To identify users who share the target user's likes and dislikes similarity measurements like cosine similarity or Pearson correlation coefficient are employed. The algorithm then suggests products that the target user is yet to interact with but that similar users have found interesting.

   Assume that a particular matrix A comprises user and item identifiers as well as ratings for films.



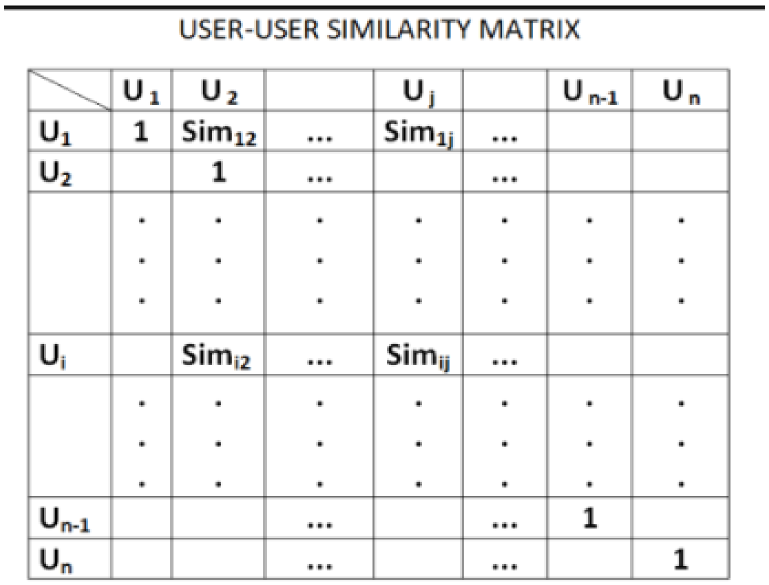
	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	4
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	4

Fig.1.4-user-items/movies rating matrix

"Source**:**https://editor.analyticsvidhya.com/uploads/687331_4VYSoN1gIgBDnQUBimudKw.png**"**

Here, Simij= similarity(useri , userj)

In here similarity can be computed through any method like cosine similarity, pearson similarity or jaccard similarity etc.



USER-USER SIMILARITY MATRIX

	U_1	U_2		U_j		U_{n-1}	U_n
U_1	1	Sim_{12}	...	Sim_{1j}	...		
U_2		1	...		...		
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
U_i		Sim_{i2}	...	Sim_{ij}	...		
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
U_{n-1}			...		...	1	
U_n			...		...		1

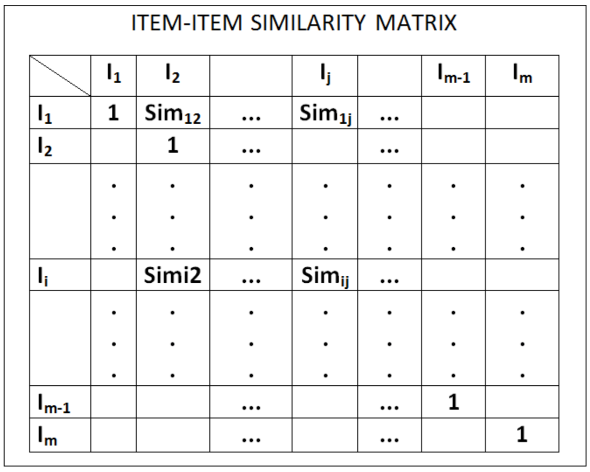
Fig.1.5- user-user matrix with Simij

"Source: https://editor.analyticsvidhya.com/uploads/71017Untitled.png"

2. **Item-Based Collaborative Filtering:**

In this method, similarities between items are calculated based on their past interactions with users, such as ratings or purchases. The algorithm then makes suggestions for products that are comparable to ones the target consumer has previously appreciated. This strategy is helpful when there are many more objects than users in the system since it may be more effective to compare similarities among items rather than between individuals.

Here,Simij= similarity(useri , userj)



ITEM-ITEM SIMILARITY MATRIX

	I_1	I_2		I_j		I_{m-1}	I_m
I_1	1	Sim_{12}	...	Sim_{1j}	...		
I_2		1	...		...		
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
I_i		Simi2	...	Sim_{ij}	...		
	⋮	⋮	⋮	⋮	⋮	⋮	⋮
I_{m-1}			...		...	1	
I_m			...		...		1

Fig.1.6- item-item/movie-movie matrix with Simij

"Source: https://editor.analyticsvidhya.com/uploads/69012ezgif-7-62fd592648.png"

Collaborative filtering techniques based on items or users each have benefits and drawbacks of their own. User-based methods can be more personalized but suffer from the "sparsity problem," where users may have only rated or reviewed a few items, making it difficult to find similar users. However, because there are typically many more items than users, item-based methods can be more effective and reliable. They may, however, be less individualised because they focus more on recommending products that are comparable to what the user has previously dealt with than on discovering new products that the user might not be aware of.

There are several advantages and disadvantages of using collaborative methods, some of which are outlined below:

Advantages:

- Diversity of perspectives: Collaborative methods can bring together people with different backgrounds, experiences, and expertise, which can lead to a broader range of perspectives and ideas.
- Increased creativity and innovation: Collaboration can encourage more creative thinking and lead to new and innovative solutions that may not have been possible with individual effort.
- Shared workload: Working collaboratively can help distribute the workload, allowing individuals to focus on their strengths and areas of expertise.
- Better decision-making: Collaborative methods can help in making more informed and well-rounded decisions by considering multiple viewpoints and opinions.
- Increased accountability: Collaboration can promote accountability among team members, as each person is responsible for their contribution towards achieving the common goal.

Disadvantages:

- Time-consuming: Collaboration can take longer than working alone, as it involves coordinating schedules, communicating, and building consensus.
- Conflicts and disagreements: Collaboration can also lead to conflicts and disagreements due to differences in opinion, values, and priorities.
- Groupthink: Collaborative methods can sometimes lead to groupthink, where members of the group conform to the dominant viewpoint or opinion, even if it is not the best decision.

4) Hybrid Based Filtering:

A recommendation system technique called hybrid-based filtering combines many different algorithms for recommendation to increase the precision and breadth of the suggestions. It combines the benefits of collaborative filtering with content-based filtering.

In a hybrid-based filtering approach, the system first uses a collaborative filtering algorithm to identify similar users and items based on their past behaviors and preferences. Then, it incorporates content-based filtering techniques to enhance the recommendations by considering the attributes and characteristics of the items.

The advantages of hybrid-based filtering include:

a) Improved accuracy: By combining multiple algorithms, the system can provide more accurate recommendations compared to individual algorithms.
b) Increased coverage: Hybrid-based filtering can help in recommending items that may not have been seen by the user before, thereby increasing the system's coverage.
c) Overcoming data sparsity: Hybrid-based filtering can handle sparse data better than individual algorithms by incorporating additional information from different sources.
d) Better handling of the cold-start problem: Hybrid-based filtering can address the cold-start problem, where new items or users have limited or no data, by leveraging the content-based approach.

However, there are also some potential drawbacks to hybrid-based filtering, including:

a) Complexity: Combining multiple algorithms can make the system more complex and difficult to implement and maintain.
b) Computational overhead: The use of multiple algorithms can increase the computational overhead, leading to slower performance.
c) Integration challenges: Integrating multiple algorithms into a single system can pose challenges in terms of data handling, model integration,

and algorithm tuning.

Overall, hybrid-based filtering can be an effective strategy for enhancing the precision and scope of recommendation systems, but it necessitates thorough analysis of the risks and difficulties involved.

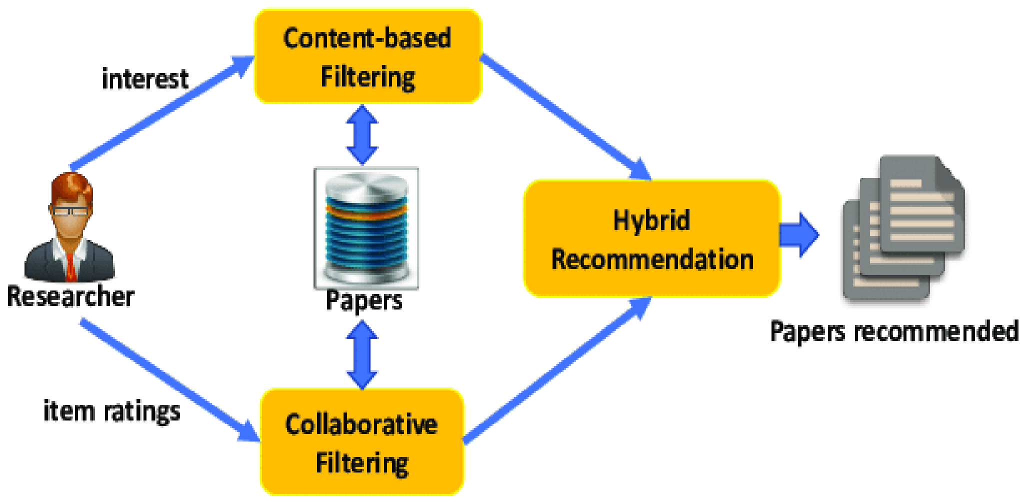


Fig.1.7-Illustration of Hybrid Filtering from [11]

5) Association Rule Mining:

A data mining approach called association rule mining involves identifying connections between the elements in a sizable dataset. Finding patterns and correlations between various objects in a dataset is frequently utilised in recommendation systems, which then produce suggestions based on these associations.

In recommendation systems, association rule mining is used to identify items that are frequently purchased or viewed together by customers. These associations can be used to generate personalized recommendations for individual customers based on their past preferences and behavior.

Analysis of transaction data is used in association rule mining to find frequent itemsets, or collections of items that are frequently seen or purchased together. The association rules that explain the relationships between the various items in the dataset may then be produced using these frequent item sets.

For example, an association rule might state that customers who purchase coffee and sugar are likely to also purchase cream. This rule could then be used to generate recommendations for customers who have purchased coffee and sugar, suggesting that they also purchase cream.

Overall, association rule mining is an important tool for building effective recommendation systems that can provide personalized recommendations to users based on their past behavior and preferences.

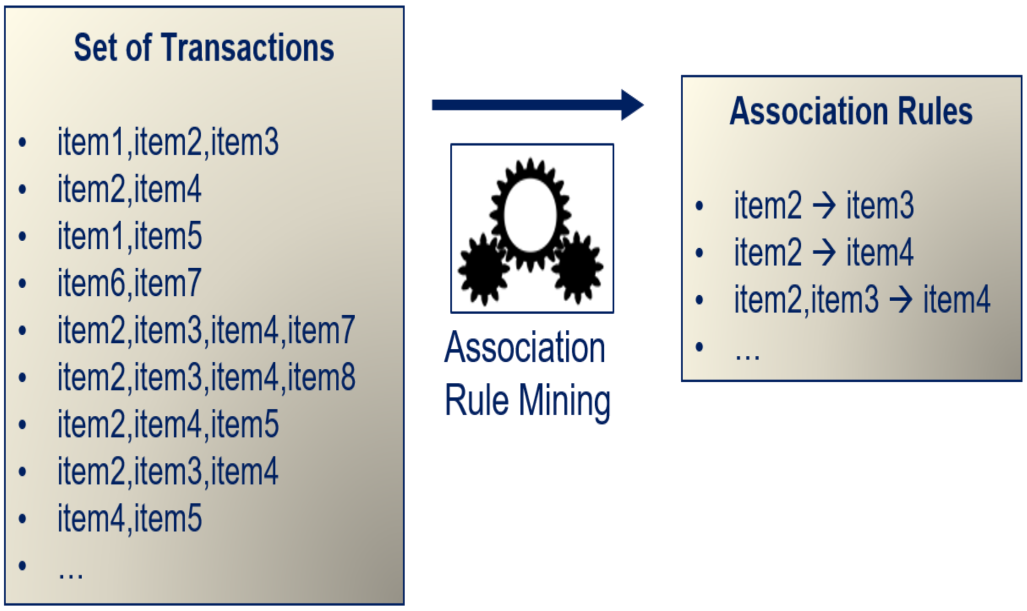


Fig.1.8 Example of Association Rule Mining

“Source: https://sherbold.github.io/intro-to-data-science/images/associationsrules_abstract.png”

A variety of association rule mining techniques exist, each with unique algorithms and methodologies. Here are a few of the most popular categories of association rule mining techniques and the corresponding algorithms.

5.1) Apriori Algorithm:

The Apriori algorithm is one among the most popular algorithms used for association rule mining. It is a bottom-up approach that works by iteratively generating candidateitemsets and pruning those failed to meet the minimum support threshold.

The algorithm is based on the following concepts:

a) Support: The support of an itemset is computed as the number of transactions in which it appears.

a) Candidate itemsets: A candidate itemset is a set of items that may be frequent. The algorithm generates these by combining smaller frequent itemsets.

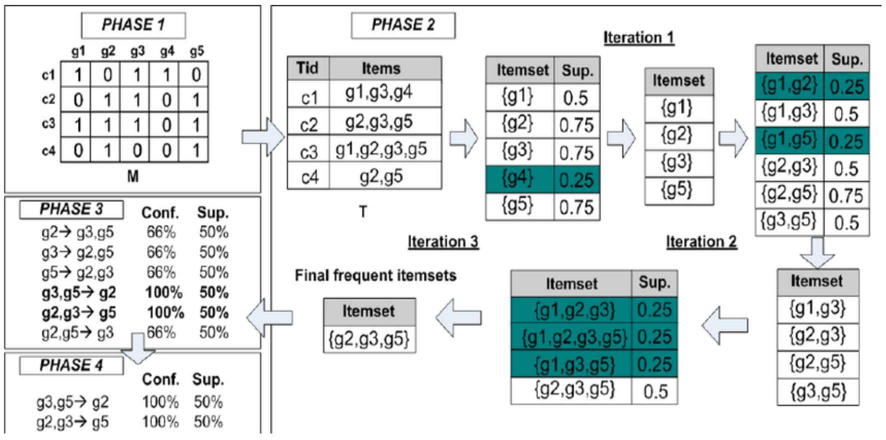
b) Frequent itemsets: A frequent itemset is a set of items that appears in a sufficient number of transactions.

The Apriori algorithm works as follows:

a) Generate frequent 1-itemsets: Scan the dataset to count the frequency of each item. Keep only the items that meet the minimum support threshold.

b) Generate frequent k-itemsets: Generate candidate k-itemsets by joining frequent (k-1)-itemsets. Prune any candidate itemsets that contain subsets that are not frequent.

c) Repeat step b until no more frequent itemsets can be generated.



	g1	g2	g3	g4	g5
c1	1	0	1	1	0
c2	0	1	1	0	1
c3	1	1	1	0	1
c4	0	1	0	0	1

Tid	Items
c1	g1,g3,g4
c2	g2,g3,g5
c3	g1,g2,g3,g5
c4	g2,g5

Itemset	Sup.
{g1}	0.5
{g2}	0.75
{g3}	0.75
{g4}	0.25
{g5}	0.75

Itemset
{g1}
{g2}
{g3}
{g5}

Itemset	Sup.
{g1,g2}	0.25
{g1,g3}	0.5
{g1,g5}	0.25
{g2,g3}	0.5
{g2,g5}	0.75
{g3,g5}	0.5

Itemset
{g1,g3}
{g2,g3}
{g2,g5}
{g3,g5}

Itemset	Sup.
{g1,g2,g3}	0.25
{g1,g2,g3,g5}	0.25
{g1,g3,g5}	0.25
{g2,g3,g5}	0.5

Itemset
{g2,g3,g5}

PHASE 3	Conf.	Sup.
g2→ g3,g5	66%	50%
g3→ g2,g5	66%	50%
g5→ g2,g3	66%	50%
g3,g5→ g2	**100%**	**50%**
g2,g3→ g5	**100%**	**50%**
g2,g5→ g3	66%	50%

PHASE 4	Conf.	Sup.
g3,g5→ g2	100%	50%
g2,g3→ g5	100%	50%

Fig.1.9- Example of Apriori algorithm from [13]

The Apriori algorithm is computationally expensive because it requires multiple passes over the dataset. However, it is effective for datasets with a relatively small number of items and a large number of transactions.

5.2) FP-Growth Algorithm:

The FP-Growth algorithm is another popular algorithm used for association rule mining. It is a divide-and-conquer approach that builds a tree structure to represent the frequent itemsets in the dataset.

The algorithm is based on the following concepts:

- Frequent itemsets: A group of goods that appear in a significant number of transactions is considered a frequent itemset.
- FP-tree: The frequent item sets in the dataset are represented by the FP-tree, a small data structure.
- Conditional pattern base: All transactions that contain a particular item are included in the conditional pattern base, a subset of the dataset.

The FP-Growth algorithm works as follows:

a) Generate frequent 1-itemsets: To determine the frequency of each item, scan the dataset. Keep just what meets the support requirement for minimal retention.
b) Build the FP-tree: To create the FP-tree, rescan the dataset. Each transaction is added to the tree, and each item's support count is updated as the tree is constructed.
c) Generate frequent itemsets: To produce frequently occurring itemsets, traverse the FP-tree. Working your way up, begin with the item that has the fewest supporters. Create a new FP-tree and the conditional pattern basis for each item. Use the new FP-tree to recursively build common itemsets.

The FP-Growth algorithm is more efficient than Apriori because it only requires two passes over the dataset. It is particularly effective for datasets with a large number of items and a small number of transactions. However, it requires more memory than Apriori because it builds a tree structure.

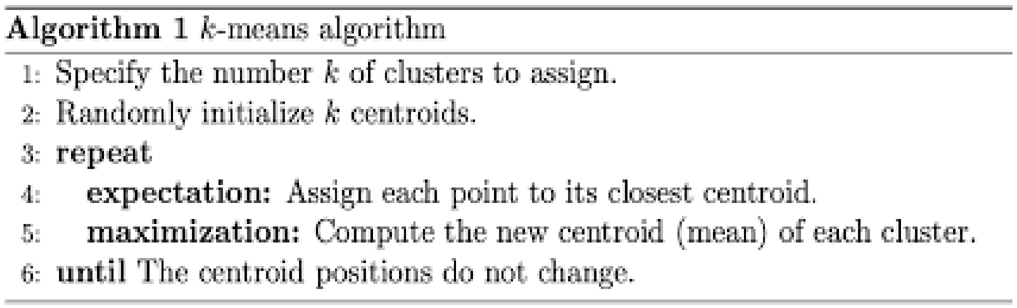
6) Clustering:

Clustering is a technique used in machine learning and data mining to group similar objects or data points together based on their attributes or characteristics. In a recommendation system, clustering can be used to group similar items or users together, and recommend items to users based on the behavior of users in the same cluster.

There are several types of clustering algorithms, including:

a. K-means clustering: It is an established clustering approach that divides data into K clusters, with each cluster determined by its centroid. The technique seeks to minimise the sum of squared distances between data points and cluster centroids.

Algorithm for the K-means clustering-



```
Algorithm 1 $k$-means algorithm
1: Specify the number $k$ of clusters to assign.
2: Randomly initialize $k$ centroids.
3: repeat
4:    expectation: Assign each point to its closest centroid.
5:    maximization: Compute the new centroid (mean) of each cluster.
6: until The centroid positions do not change.
```

Fig.1.10-Algorithm for k-means

“Source: https://files.realpython.com/media/kmeans-algorithm.a94498a7ecd2.png”

b. Hierarchical clustering: This method organises data points into a dendrogram-like hierarchy of clusters. There are two different types of algorithms: agglomerative, which begins with each point of data and clusters them, and divisive, which begins with all the data pieces in one cluster and repeatedly divides them into smaller clusters.

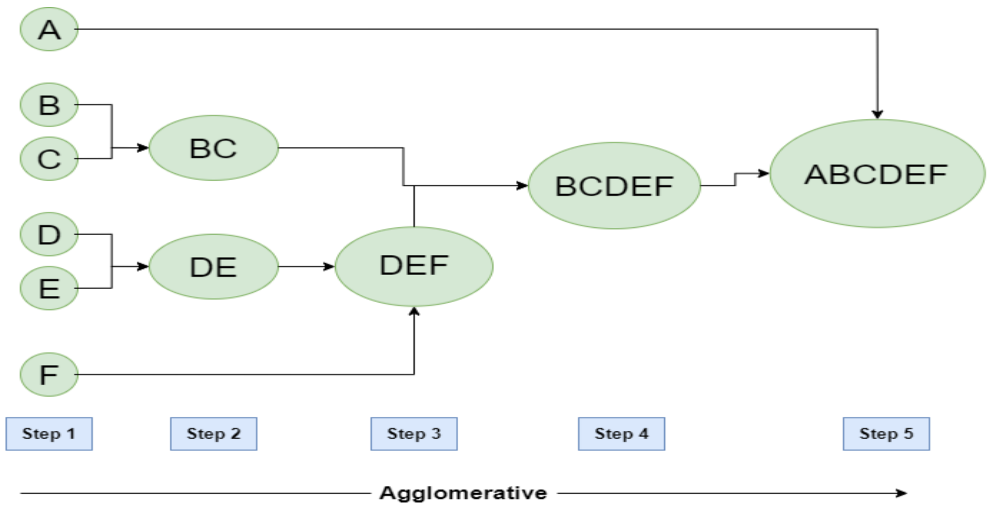


Fig.1.11-Illustration of Hierarchical clustering

"Source: https://media.geeksforgeeks.org/wp-content/uploads/20200204181551/Untitled-Diagram71.png"

c. Density-based clustering: This algorithm groups data points based on the density of the data points in the data space. Data points that are closer together are considered to be in the same cluster.

To use clustering in a recommendation system,the steps are:

a) Collect data on user behavior or item attributes.

b) Preprocess and clean the data.

c) Choose a clustering algorithm such as K-means, hierarchical clustering, or density-based clustering.

d) Apply the clustering algorithm to group similar items or users together based on their attributes or behavior.

e) Make recommendations by recommending items that are frequently purchased or rated highly by users in the same cluster.

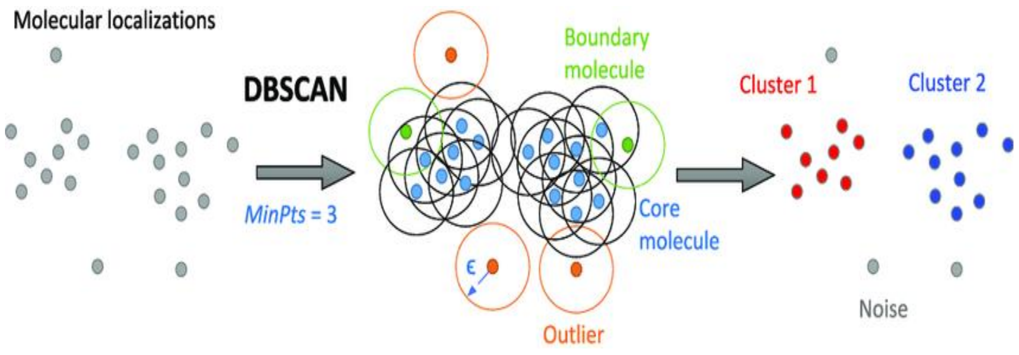


Fig.1.12-Illustration of Density-based clustering from [12]

This is a high-level overview, and there may be additional steps involved in implementing a clustering-based recommendation system, such as evaluating the quality of the clusters and recommendations, refining the model, and deploying the final model in a production environment.

It's important to note that clustering is just one approach to recommendations and it may not work for all types of data or recommendation scenarios. Other approaches, such as collaborative filtering, content-based filtering, and matrix factorization, can be used alongside with clustering to increase recommendation accuracy and significance.

Some general advantages and disadvantages of using clustering in a recommendation system.

Advantages:

- Personalization: Clustering can group similar items or users based on their characteristics or preferences, enabling personalized recommendations.
- Scalability: Clustering can handle large datasets, making it suitable for recommendation systems that have many items or users.
- Diversity: Clustering can ensure diversity in recommendations by identifying clusters of items or users that are different from each other.
- Real-time recommendations: Clustering can quickly identify similar items or users, making it possible to generate recommendations in real-time.

Disadvantages:

- Cold start problem: Clustering requires a certain amount of data to group items or users, making it challenging to provide recommendations for new or less popular items.
- Lack of interpretability: The clustering process may not provide insights into why certain items or users are grouped together, making it difficult to explain recommendations to users.
- Similarity metrics: The accuracy of clustering depends on the choice of similarity metrics, which may not always capture the complexity of user preferences or item characteristics.
- Over-specialization: Clustering can result in over-specialization, where recommendations are limited to a small subset of items or users.

These methodologies can be combined or customized to suit the specific requirements of the recommendation system. The choice of methodology depends on the type of data available, the size of the dataset, and the desired level of recommendation accuracy.

Major Challenges to the Recommendation system:

1) Cold start problem:

There might not be enough information available when a new user or item is introduced to the system for accurate suggestions. This is known as the "cold start problem," and collaborative filtering algorithms may find it particularly difficult. The system can employ collaborative filtering, content-based approaches, or suggestions based on content to alleviate the cold start issue.

2) Data sparsity:

Recommendation systems rely on user behaviour data to provide suggestions, however this data is frequently lacking, particularly for specialised or newly released products. As a result, it could be challenging to come up with reliable suggestions for these things. The system can leverage contextual data or methods like matrix factorization or feature engineering to employ partial data to overcome data sparsity.

3) Scalability:

The processing resources needed to make suggestions might become prohibitive as a system's user base and inventory expand. Long processing times or the requirement for distributed computing solutions may result from this. The system can make advantage of model parallelism or data parallelism, as well as distributed computing frameworks like Hadoop or Spark, to overcome scalability issues.

4) Privacy concerns:

Recommendation systems frequently rely on user data collection, which might cause privacy issues. If users are unsure that their information will be protected and handled appropriately, they may be reluctant to provide it.The system can put robust data security measures in place and offer detailed explanations of how user data is gathered and utilised to satisfy privacy concerns.

5) Diversity and novelty:

Given that they frequently base their recommendations on past user behaviour, recommendation systems may find it difficult to suggest unique or diverse items. This can lead to the users' recommendations being monotonous. The system can employ strategies like diversity-aware recommendation or serendipity-based recommendation to overcome issues with novelty and diversity.

6) Bias:

In particular, biassed data used to train the system can make recommendation systems biassed. This may lead to suggestions that favour particular groups or uphold current disparities.The system can employ debiasing, fairness-aware recommendation, or explainable recommendation strategies to solve bias concerns.

7) Explanation and transparency

Users may find it challenging to comprehend how recommendations are made since recommendation systems can be complicated and opaque. The system can allow users control over the suggestion process and comprehensive explanations of how recommendations are made in order to solve the difficulties of explanation and openness.

CHAPTER-2

LITERATURE SURVEY

Recommendation systems are widely used in various domains, including e-commerce, social media, and entertainment. They are used to suggest items or content to users based on their preferences and past behavior. Biclustering, collaborative filtering, and hybrid approaches are some of the most commonly used techniques for building recommendation systems and our main focus will be on biclustering method.

Biclustering is a data mining technique that aims to simultaneously cluster rows and columns of a data matrix. In other words, it partitions both the rows and columns of a dataset into groups such that the data within each group exhibit similar patterns or characteristics. Biclustering is also known as co-clustering or bi-clustering.

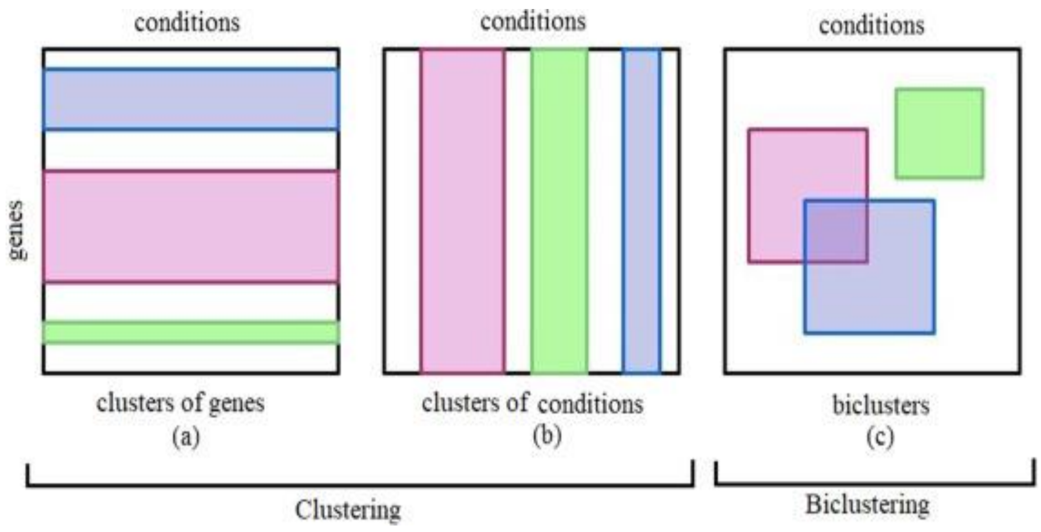


Fig.2.1-Illustration of biclustering from [9]

Biclustering can be applied to various types of data, such as gene expression data, text data, image data, and user-item interaction data. It is particularly useful in domains where the data have multiple attributes or dimensions and exhibit complex patterns of dependencies.

Biclustering algorithms can be classified into two main categories: deterministic and probabilistic. Deterministic biclustering algorithms aim to find a fixed number of biclusters that optimize a predefined objective function, whereas probabilistic biclustering algorithms aim to estimate the probability of each data point belonging to a bicluster. Examples of deterministic biclustering algorithms include Spectral Co-clustering, Bimax, and Plaid, whereas examples of probabilistic biclustering algorithms include Bayesian biclustering and Bayesian co-clustering.

Biclustering has several applications, including recommendation systems, image analysis, gene expression analysis, and text mining. In the context of recommendation systems, biclustering can be used to identify groups of users with similar preferences and groups of items with similar attributes, which can be used to generate personalized recommendations.

An overview of the literature on building recommendation systems with biclustering is given in this section. Each approach is explained in detail along with its advantages and disadvantages in respect to the emphasised problems. The literature reports several biclustering-based recommendation systems, but none of them address privacy issues. The many definitional decisions taken for biclusters and biclustering, which vary greatly, are shown in Fig. 14.The different methods either consider extensive biclustering (light purple cell), global optimisation objectives (purple cells), or local limitations (blue cells). Some research investigations use a combination of regional and global restrictions. Others adopt a modular approach and use renowned bioinformatics community algorithms, while some papers create their own biclustering algorithms to address particular problems.

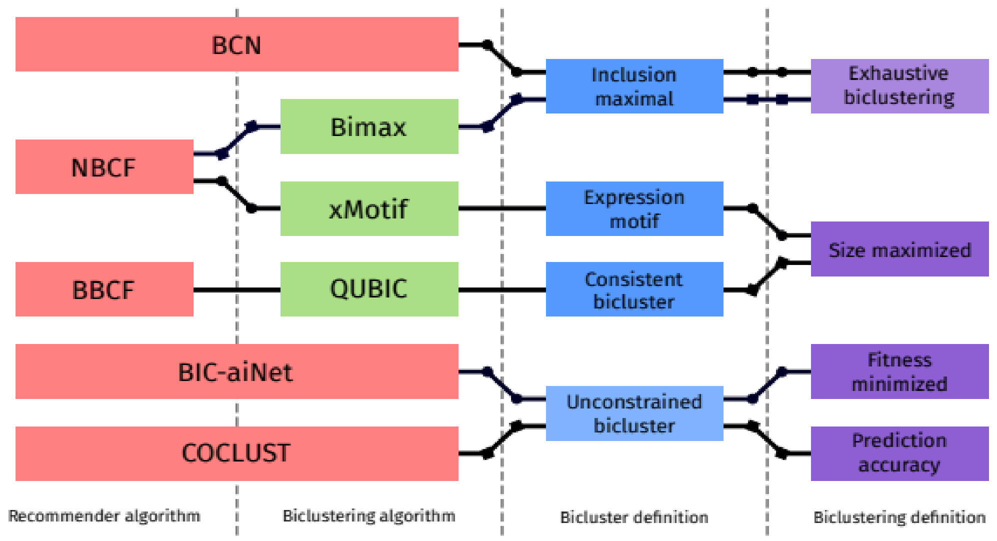


Fig.2.2-Illustration of types of approaches from [8]

Figure 2.2: Outline of several biclustering-based algorithms for recommendations with respect to the restrictions taken into account.

a) COCLUST:

The paper **[3]** proposes a biclustering method used for collaborative filtering, which offers an alternative to model-based techniques that are computationally expensive but highly effective. The approach uses a weighted Bregmanbi clustering algorithm from **[2]** to construct a partitional biclustering, which provides a locally optimal solution. Unlike standard biclustering methods, this approach considers the biases introduced by the specific user and item is being evaluated to generate more precise predictions. The algorithm begins with random partitions and iteratively optimizes the assignments of rows and columns until a convergence criterion is met. The overall number of biclusters stays constant throughout the procedure, while input factors like the amount of item groupings and user clusters in the checkerboard layout are supplied. However, this constraint could become problematic in the long run if the amount of data increases significantly. To address this issue, the authors propose an incremental training method for real-time systems where new ratings may not initially be assigned to a bicluster. A parallel version of the algorithm is also presented to enhance its scalability, but it requires a good partitioning schema to avoid unbalancing the load between processes. The experimental evaluation conducted on several real-world datasets demonstrates that the proposed COCLUST approach provides high-quality recommendations in terms of MAE score while outperforming 14 state-of-the-art model-based approaches in terms of time performance. Although the paper offers some guidance on selecting input parameters, a more extensive analysis is required to determine their optimal values for practical systems.

b) NBCF:

The authors of **[7]** suggest NBCF, a novel algorithm for collaborative filtering. NBCF biclusters the user-item matrix using known methods, such as Bimax**[5]** for binary ratings or xMotif**[4]** for size-maximized expression motifs. NBCF employs a similarity measure to find the k closest biclusters to a specific user and then uses them to forecast unknown item ratings for that user. The authors perform a thorough experimental investigation of the effects of bicluster sizes and the value of k on precision/recall findings and average recommendation time. On real-world datasets, In regards to recommendation quality and time efficiency, NBCF surpasses cutting-edge user- and item-based techniques, but not when measured against well-liked model-based methods like matrix factorization. The flexibility of NBCF enables for easy integration of various biclustering algorithms, however the influence of biclustering method selection on recommendation performance warrants additional investigation. The authors notice better results with xMotif than with Bimax, but conclude that a comprehensive examination is required due to the latter's action on binary data. It is unknown if NBCF suffers from the scalability concerns that plague neighborhood-based techniques that need several similarity calculations.

c) BCN :

The BCN approach proposed in **[1]** uses an exhaustive biclustering and a complete lattice to recommend items to users. It starts by finding the smallest bicluster that contains the target user and explores the neighborhood biclusters to identify candidate items, which are then ranked by computing a similarity-based score. The authors claim that BCN performs better than existing state-of-the-art algorithms in terms of recommendation quality, especially for sparse data, and is efficient in terms of scalability. It also has the advantage of not requiring re-training of the model when new ratings appear, making it suitable for dynamic settings, and does not need parameter tuning. However, BCN works with binary data, and a discretization step is required for non-binary ratings, which can result in a loss of valuable information. It is not clear how BCN can be adapted to handle non-binary data.

d) BBCF:

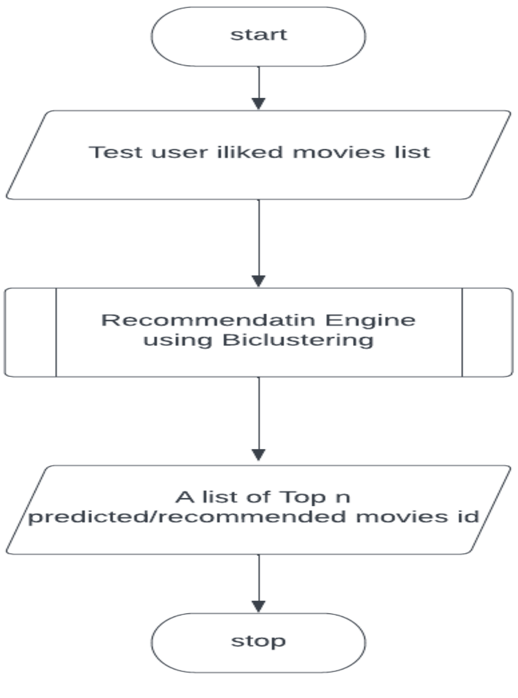
The authors of **[6]** suggest a BBCF algorithm that is based on NBCF. They begin by using QUBIC to bicluster the user-item matrix in order to locate coherent biclusters. Utilising the same resemblance criteria as NBCF, they then identify a group of k biclusters that are collectively the most comparable to the evaluated user. These biclusters are joined to create a sub-matrix, which is then used in the conventional item-based KNN technique to generate suggestions. When tested on real-world datasets, BBCF outperforms item-based and model-based techniques in terms of suggestion quality and throughput. The influence of various characteristics, such as the number of nearest neighbours, is investigated. Despite being a direct extension of NBCF's core concept, it is not compared to it. BBCF, like NBCF, is modular in terms of the biclustering algorithm utilised, which raises questions about the influence this option has on performance. The authors also emphasise that the biclustering process is often computed offline, and the quest for an appropriate approach to include fresh ratings without having to re-run the entire biclustering process is currently ongoing in the case of a dynamic system.

CHAPTER-3

SYSTEM DESIGN AND DEVELOPMENT

1) Dataset Used:

The "ml-latest-small" information collection, which includes free-text labelling and 5-star rating activity from MovieLens, is available online. 100836 ratings, 9742 films, and 3683 tag applications are included. Between March 29, 1996, and September 24, 2018, 610 users created the data included in these entries. Creating this dataset took place on September 26, 2018. Users were chosen at random to be taken into account. Each of the selected users has evaluated at least 20 films.When creating the model, we restricted our evaluation to the top 2000 movie IDs.This information is a portion of a larger dataset that comprises files with 26 million ratings for all 45,000 films from 270,000 different people. The evaluations, which range from 1 to 5, came from the GroupLens website.


start
Test user iliked movies list
Recommendatin Engine using Biclustering
A list of Top n predicted/recommended movies id
stop

.

Fig.3.1-Flowchart of a general recommendation system

2) Method:

The development of a recommendation system involves several steps, including data collection, preprocessing, and model training. In this section, we describe the steps involved in developing a recommendation system that utilizes a user-item matrix and matrix factorization approach. The steps are:

- **Data Collection:** Data on user interactions with objects must first be gathered. Data about user reviews, sales, clicks, views, and other interactions with products can be included. Online marketplaces, social networking sites, and e-commerce websites are just a few of the areas where the data may be gathered.
- **Preprocessing:** The data must be preprocessed after collection in order to be used in the recommendation system. In this stage, we create a user-item matrix from the raw data. The matrix, where each row denotes a person and each column is an item, provides data about how people interacted with the things. The mean of the ratings is then subtracted from each rating to normalise the user-item matrix and correct for any biases in the data.
- **Matrix Factorization:** The user-items matrix is split into two lower-dimensional matrices, the user-user matrix and the items-items matrix, in the next phase using matrix factorization. In contrast to the item-item matrix, which includes latent properties of the items such as similarity between the items, the users-users matrix contains latent aspects of the users such as similarity between the users. The user and item matrices are then normalised to make sure that their values fall within a particular range.
- **Prediction/ Recommendations:** Once the user and item matrices are computed, we can use them to predict the preferences of users for items they have not yet interacted with. This is done by using the clustering algorithm on the users-users matrix and items-items matrix and then further by finding similarity among the users in a cluster and finally using the similar score among the similar user for the movie/items we recommend the top-n movies.

- **Model Evaluation:** Finally, we evaluate the performance of the recommendation system using various metrics, such as confusion matrix, precision, recall, and mean average precision. These metrics help us to determine the effectiveness of the recommendation system and identify areas for improvement.

In summary, the development of a recommendation system involves collecting and preprocessing data, performing matrix factorization, predicting user preferences, and evaluating the performance of the system.

Explaning the steps in brief:

a) Data collection:

We start from importing the data from the source which in this case will contain around 600 users and 2000 movies.

b) Preprocessing

Once we have our dataset we perform the following steps:

- Convert the dataset into a user vs movie ratings matrix where each row represents movies such that each user and each column represent a unique movie id mapped to each movie and finally the cell represents the ratings.
- Fill the empty cells of the matrix with 0.

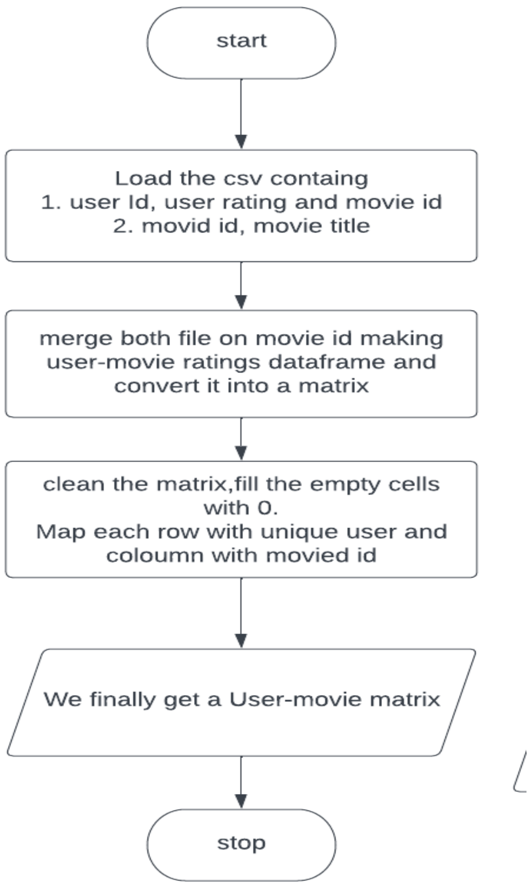

start
Load the csv containg
1. user Id, user rating and movie id
2. movid id, movie title
merge both file on movie id making user-movie ratings dataframe and convert it into a matrix
clean the matrix,fill the empty cells with 0.
Map each row with unique user and coloumn with movied id
We finally get a User-movie matrix
stop

Fig.3.2-Flowchart of preprocessing steps

c) Matrix factorization:

Once we have the user movie matrix we perform the following steps:

1. From the user-movie matrix create 2 different matrix namely user-user matrix and movie-movie matrix by following the steps:
    a. Find the transpose of the user- movie matrix which is movie-user matrix.
    b. Now to create a user-user matrix find the dot product of the users-movies matrix and movies- users matrix.
    c. Similarly, to create a movie-movie matrix find dot product of movie- user matrix and user-movie matrix.
    d. Now we have both the matrix with us which represent correlation as cell value.

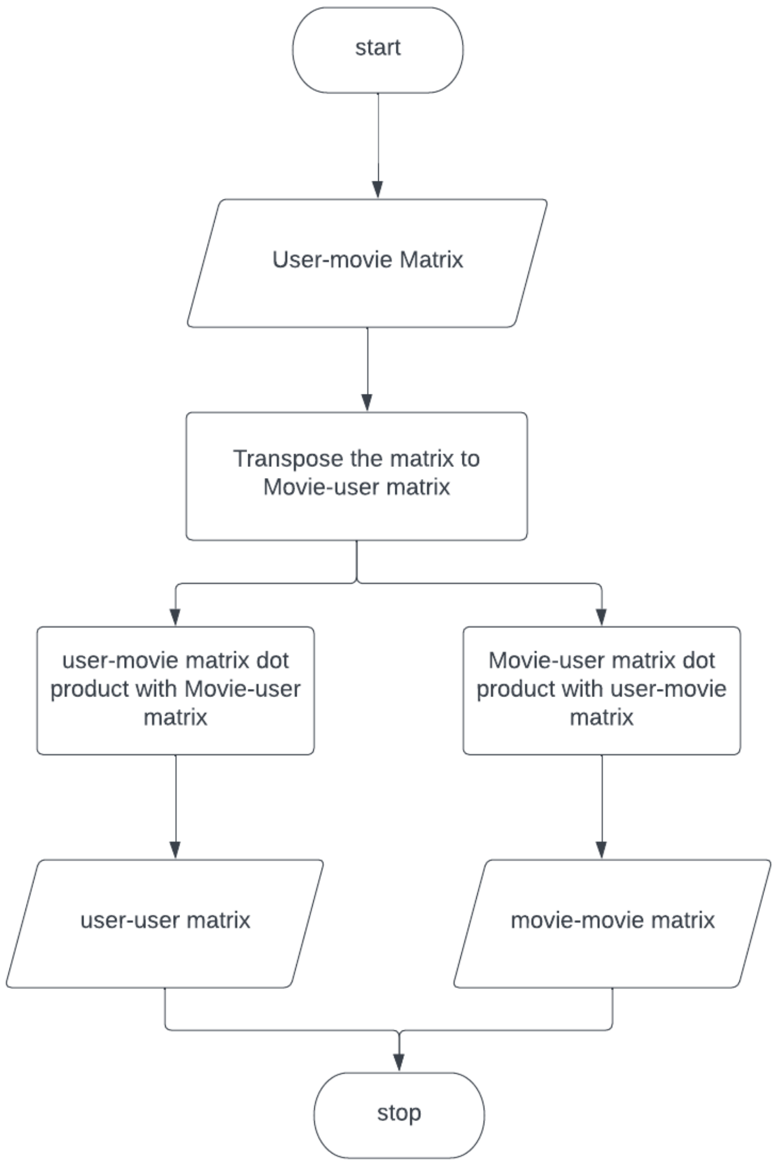

start
User-movie Matrix
Transpose the matrix to Movie-user matrix
user-movie matrix dot product with Movie-user matrix
Movie-user matrix dot product with user-movie matrix
user-user matrix
movie-movie matrix
stop

Fig.3.3-Flowchart of matrix factorization

2. Normalize the obtained matrices.

d) Prediction/ Recommendations:

Now we have 3 matrices with us namely user-movie matrix, user-user matrix and movie-movie matrix.

Now to provide recommendations we have to select a target user for whom the recommendation is to be made for testing purposes and to do so we can follow the steps.

1. From the original matrix user-movie matrix select a random row which represents the target user.
2. Now in order for the system to recommend the movie as output it must need some input to process on. So we need some test input from the user and to get input follow the step:
    a. Select the target user row.
    b. Find all the movies represented by the column no. which are having value as 1.
    c. Now of all the found movie ids randomly change some of the movie ids value to 0 and name these sets of movies as “changed movies”.
    d. Now the remaining movies ids with value 1 will be taken as the test inputs for the recommendation system and those with the value 0 are to be recommended by the system.

Now that we have our target user and test input we need a model which take both of these parameter as input and procure recommend movie and to do so follow the below steps:

On user-user matrix-

1. Now apply K means clustering on both user-user matrices.
2. Now we have a cluster of similar users represented by row numbers.
3. Now for each cluster (containing multiple users each) find the common movies ids watched and rated by the entire cluster.
4. Now once we have found the common movie ids for all clusters save those both user clusters and corresponding common movies in a csv let it be “user-movie-clustered.csv” with headers as “user_list” and

"movie_list".

Now, On movie-movie matrix-

i. Now applying K means clustering on both movie-movie matrix.
ii. Now we have a cluster of similar movies represented by column numbers.
iii. Now for each cluster (containing multiple movies each) find the common users who have watched and rated all the movies in the cluster.
iv. Now once we have found the common users for all clusters, save those both user clusters and corresponding common movies "user-movie-clusterd.csv" file formed above under the corresponding headers.

Now we have the "user-movie-clustered.csv " file to find the most users to the target user for recommendation follow the steps:

i. Now we have a file counting a list of clusters of users and corresponding movies and vice versa.
ii. From each user cluster in the file under column "user_list" select the first user from each cluster and find its simmiliary from the target user.
iii. Store the similarity score of each first user in the cluster in a dictionary with key as the row no. of that user and value as the similarity score.
iv. Sort the dictionary in descending order.
v. Now select the first element from the dictionary which has the highest similarity score, extract the key representing the row no. and from the "user-movie-clustered.csv" extract all the users from the particular row.
vi. Now these extracted users are most similar to the target users.

Now we have a target user, test inputs from the target user and a list of users similar to the target user, our final task is to find the most similar movies for recommendations to the target user. Follow the steps:

i. Use a similarity measure such as cosine similarity or Pearson correlation coefficient to compute the similarity between the target user and all other users.
ii. Find the top k most similar users to the target user.
iii. Based on the ratings of these top k users, predict the ratings for the movies that the target user has not yet rated.
iv. Recommend the top n movies with the highest predicted ratings to the target user.

So now we have our Recommend n-movies for the target user.

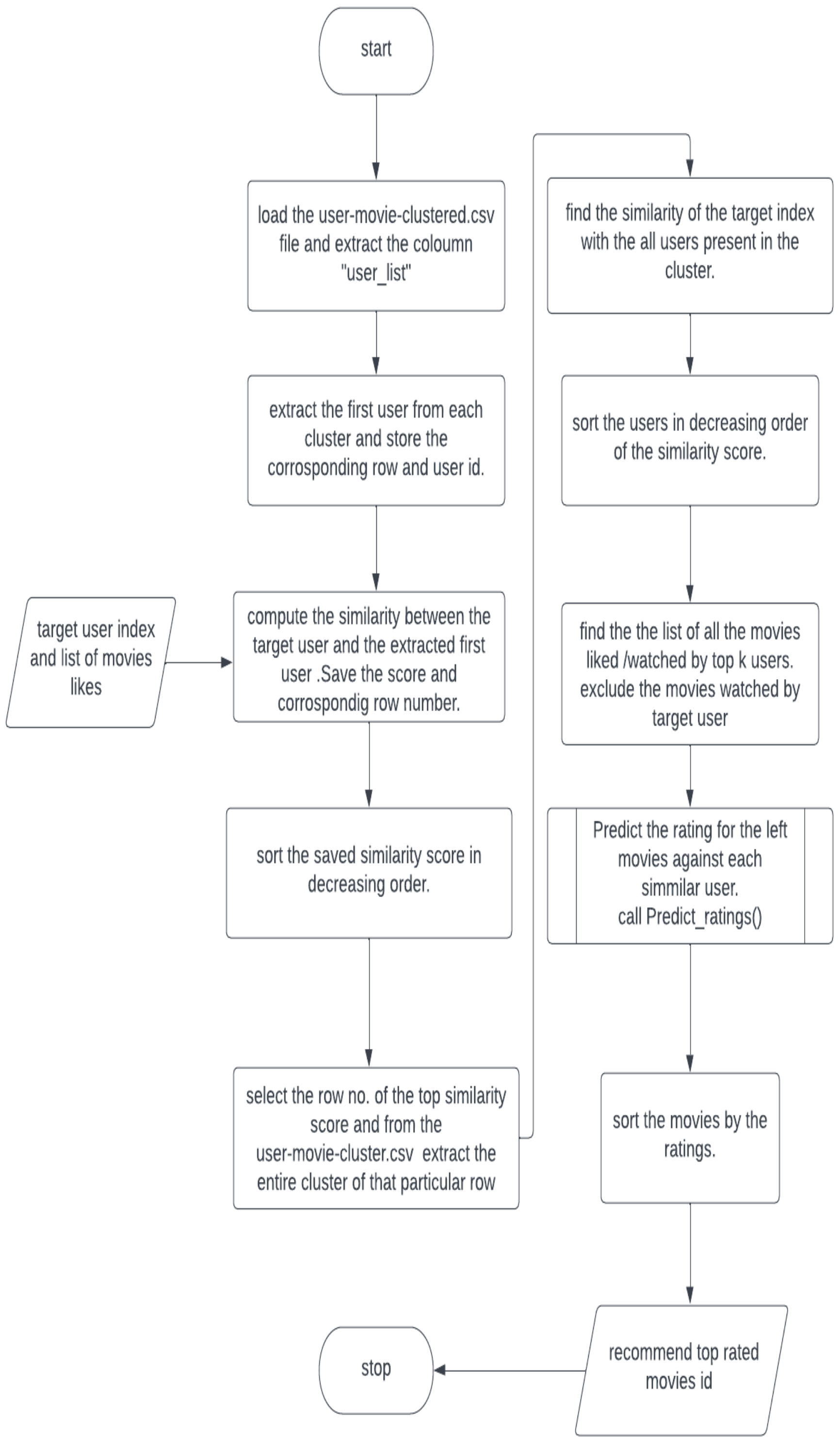

start
load the user-movie-clustered.csv file and extract the coloumn "user_list"
extract the first user from each cluster and store the corrosponding row and user id.
target user index and list of movies likes
compute the similarity between the target user and the extracted first user .Save the score and corrospondig row number.
sort the saved similarity score in decreasing order.
select the row no. of the top similarity score and from the user-movie-cluster.csv extract the entire cluster of that particular row
find the similarity of the target index with the all users present in the cluster.
sort the users in decreasing order of the similarity score.
find the the list of all the movies liked /watched by top k users. exclude the movies watched by target user
Predict the rating for the left movies against each simmilar user. call Predict_ratings()
sort the movies by the ratings.
recommend top rated movies id
stop

Fig.3.4-Flowchart for recommendation process

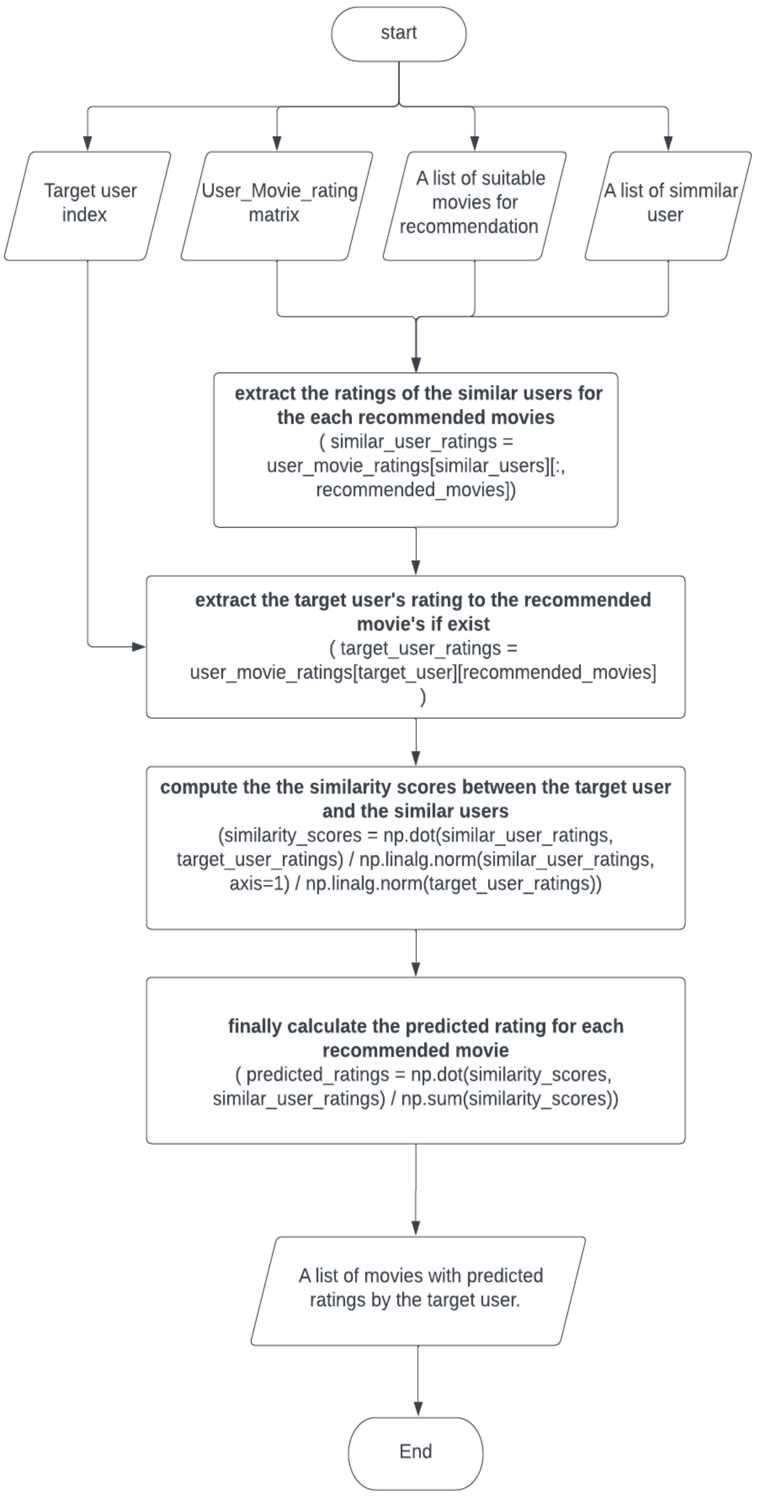

start
Target user index
User_Movie_rating matrix
A list of suitable movies for recommendation
A list of simmilar user
extract the ratings of the similar users for the each recommended movies
(similar_user_ratings = user_movie_ratings[similar_users][:, recommended_movies])
extract the target user's rating to the recommended movie's if exist
(target_user_ratings = user_movie_ratings[target_user][recommended_movies])
compute the the similarity scores between the target user and the similar users
(similarity_scores = np.dot(similar_user_ratings, target_user_ratings) / np.linalg.norm(similar_user_ratings, axis=1) / np.linalg.norm(target_user_ratings))
finally calculate the predicted rating for each recommended movie
(predicted_ratings = np.dot(similarity_scores, similar_user_ratings) / np.sum(similarity_scores))
A list of movies with predicted ratings by the target user.
End

Fig.3.5- Flowchart for rating prediction process

The formula used above to calculate the predicted ratings is a version of the weighted sum method of similarity. The formula can be broken down into two parts:

1) Similarity scores calculation:

similarity_scores=np.dot(similar_user_ratings,target_user_ratings)/
(np.linalg.norm(similar_user_ratings,axis=1)/
np.linalg.norm(target_user_ratings))

This calculates the cosine similarity between the target user and the similar users for the recommended movies. It uses the dot product of the similar user ratings and target user ratings, divided by the product of the L2 norm of the similar user ratings and the L2 norm of the target user ratings. This gives us the similarity scores for each similar user and recommended movie pair.

2) Predicted ratings calculation:

predicted_ratings=np.dot(similarity_scores,similar_user_ratings)/
np.sum(similarity_scores)

This calculates the predicted ratings for each recommended movie by taking the dot product of the similarity scores and the similar user ratings for the recommended movies, and dividing it by the sum of the similarity scores. This gives us a weighted sum of the similar user ratings, where the weights are given by the similarity scores

e) **Model evaluation:**

Now that we have our test user, test input , and recommended movies we need to evaluate the model.We have to keep in mind that along with movie recommendation if we are predicting the ratings or not.

The general steps for the model evaluation are as follows:

1. Split the dataset into train and test datasets: To evaluate the performance of the model, we first need to split the dataset into training and testing datasets. Typically, we split the dataset into 80:20 or 70:30 ratios.
2. Train the model: We need to train the recommendation model using the training dataset.
3. Generate recommendations for the test dataset: Once the model is trained, we can generate recommendations for the test dataset.
4. Evaluate the model using different metrics:
   - Find True Positive Rate(TP), False Positive Rate(FP), False Negative Rate(FN).
     - *TP=len(set(recommended_movies)&set(changed_movies)),*
     - *FP=len(recommended_movies)-TP*
     - *FN=len(changed_movies)-TP.*
   - Precision: Precision measures the proportion of recommended items that are relevant to the user. It is calculated by dividing the number of relevant items advised by the model by the total number of items recommended.
     - *Precision = TP / (TP + FP) if TP + FP> 0 else 0*
   - Recall: The proportion of relevant items proposed by the model is measured by recall. It is defined as the model's suggested number of relevant items divided by the total number of relevant items in the test dataset.
     - *Recall = TP / (TP + FN) if TP + FN> 0 else 0*

- F1-score: F1-score is the harmonic mean of precision and recall. It is a balanced measure that takes both precision and recall into account.
    - *F1 = 2 * precision * recall / (precision + recall) if precision + recall > 0 else 0*
- Accuracy: Accuracy measures the proportion of correctly predicted ratings. It is defined as the number of correctly predicted ratings divided by the total number of ratings in the test dataset.
    - *Accuracy = len(set(recommended_moviess) & set(changed movies)) / len(set(recommended_moviess) | set(changed movies))*

5. Repeat the process for different models: To compare the performance of different recommendation models, you can repeat the above process for each model and choose the model that performs best on the test dataset.

Choose the appropriate metric: You may select the ideal metric to assess the model depending on the issue and the dataset. For instance, you may leverage accuracy and recall to suggest films to a user based on their tastes. RMSE and MAE can be used to forecast movie ratings.

CHAPTER 4

EXPERIMENTS AND RESULT ANALYSIS

This section contains an experimental investigation of our suggested framework. It discusses the experimental setup, then offers the experiment outcomes before summarising our findings.

A. Dataset

We make use of GroupLens Research's MovieLens dataset. It is a dataset that is openly accessible.All 45,000 films in the Full MovieLens collection have their metadata included in the collection. The films in the dataset have a release date of July 1, 2017, or earlier. TMDB vote counts and vote averages are among the data points. Other data points include cast, crew, plot keywords, budget, revenues, banners, release dates, languages, production firms, and nations. For all 45,000 of the films in this collection, there are additional files with almost 26 million ratings from 270,000 different people. The ratings, which range from 1 to 5, came from the GroupLens website.

B. Hardware and Software

This section contains details about the hardware and software used to carry out the tests.

1) Hardware:

- Processor Type Used: Intel Core i3 CPU
- Processor Speed Used: 2.53 GHz
- Available Ram Required: 12.00 GB (vary per dataset)
- Good internet connection is must.

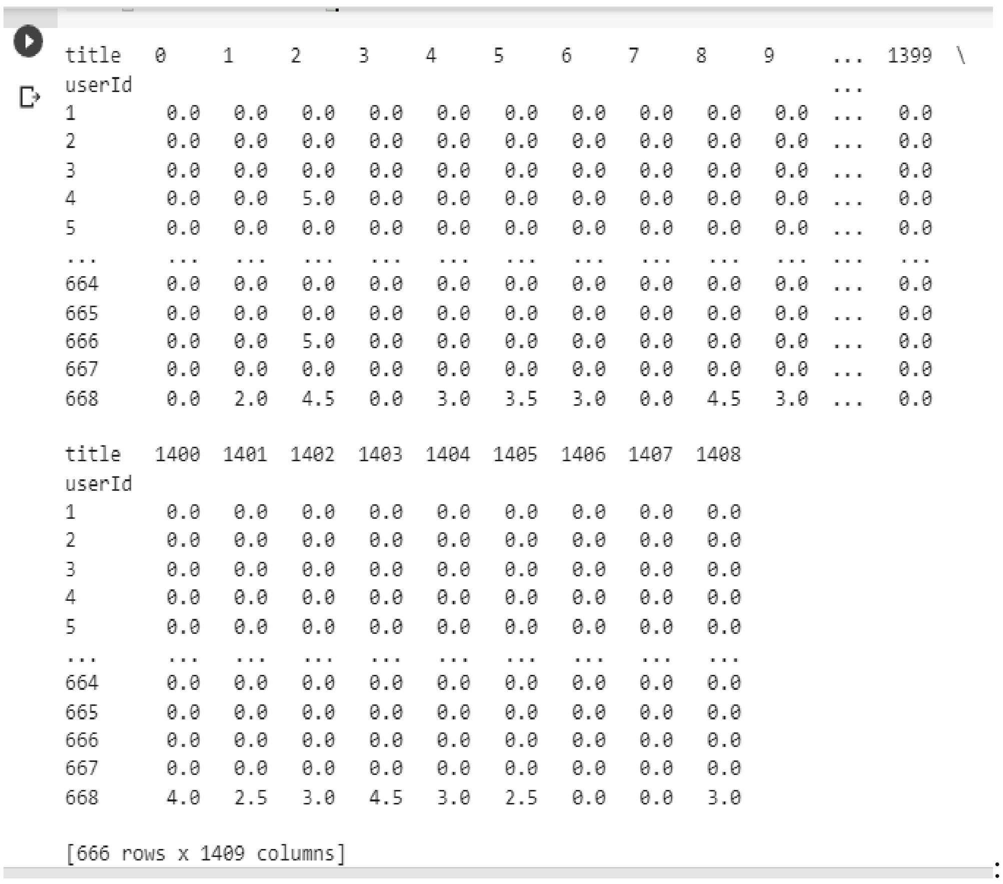
2) Software:

- Google collab or jupiter notebook.
- Python and related packages for data mining and cleaning

C. Experiments/implementation:

In this section, we show the specifics of the tests performed on the suggested solution algorithm's sections, Favourite Items and Non-Favorite Items, as well as the results obtained from those trials.

1) Importing dataset into google collab and converting it into user-movie matrix and clean the data we have:

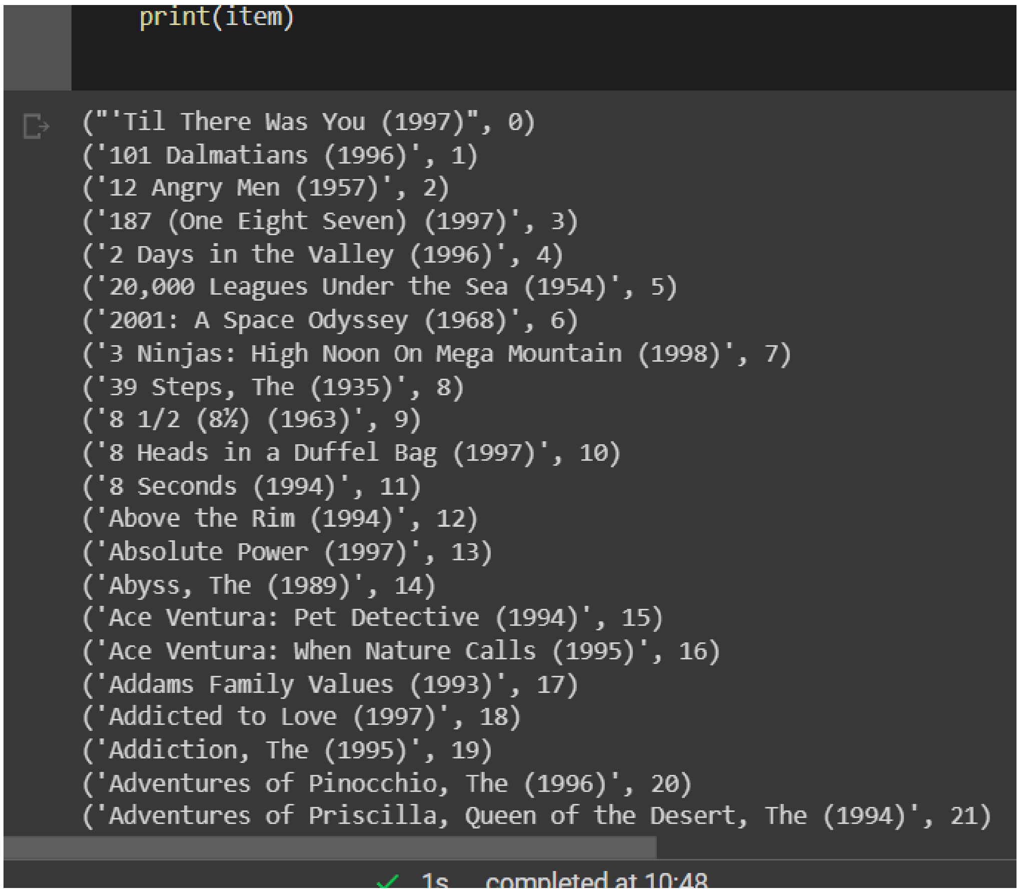


```
title   0     1     2     3     4     5     6     7     8     9     ...  1399  \
userId                                                          ...
1       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
2       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
3       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
4       0.0   0.0   5.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
5       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
...     ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
664     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
665     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
666     0.0   0.0   5.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
667     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   ...   0.0
668     0.0   2.0   4.5   0.0   3.0   3.5   3.0   0.0   4.5   3.0   ...   0.0

title   1400  1401  1402  1403  1404  1405  1406  1407  1408
userId
1       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
2       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
3       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
5       0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
...     ...   ...   ...   ...   ...   ...   ...   ...   ...
664     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
665     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
666     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
667     0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
668     4.0   2.5   3.0   4.5   3.0   2.5   0.0   0.0   3.0

[666 rows x 1409 columns]
```

:

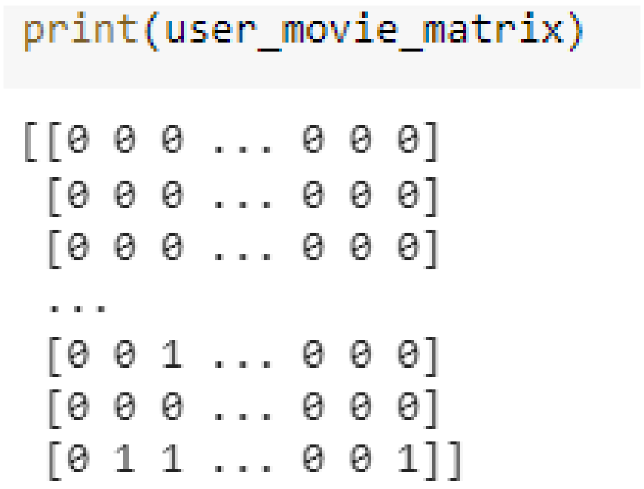
Fig.4.1-Output of user-title of movie dataframe

In here each column is mapped to a specific movie



```
    print(item)

("'Til There Was You (1997)", 0)
('101 Dalmatians (1996)', 1)
('12 Angry Men (1957)', 2)
('187 (One Eight Seven) (1997)', 3)
('2 Days in the Valley (1996)', 4)
('20,000 Leagues Under the Sea (1954)', 5)
('2001: A Space Odyssey (1968)', 6)
('3 Ninjas: High Noon On Mega Mountain (1998)', 7)
('39 Steps, The (1935)', 8)
('8 1/2 (8½) (1963)', 9)
('8 Heads in a Duffel Bag (1997)', 10)
('8 Seconds (1994)', 11)
('Above the Rim (1994)', 12)
('Absolute Power (1997)', 13)
('Abyss, The (1989)', 14)
('Ace Ventura: Pet Detective (1994)', 15)
('Ace Ventura: When Nature Calls (1995)', 16)
('Addams Family Values (1993)', 17)
('Addicted to Love (1997)', 18)
('Addiction, The (1995)', 19)
('Adventures of Pinocchio, The (1996)', 20)
('Adventures of Priscilla, Queen of the Desert, The (1994)', 21)
```

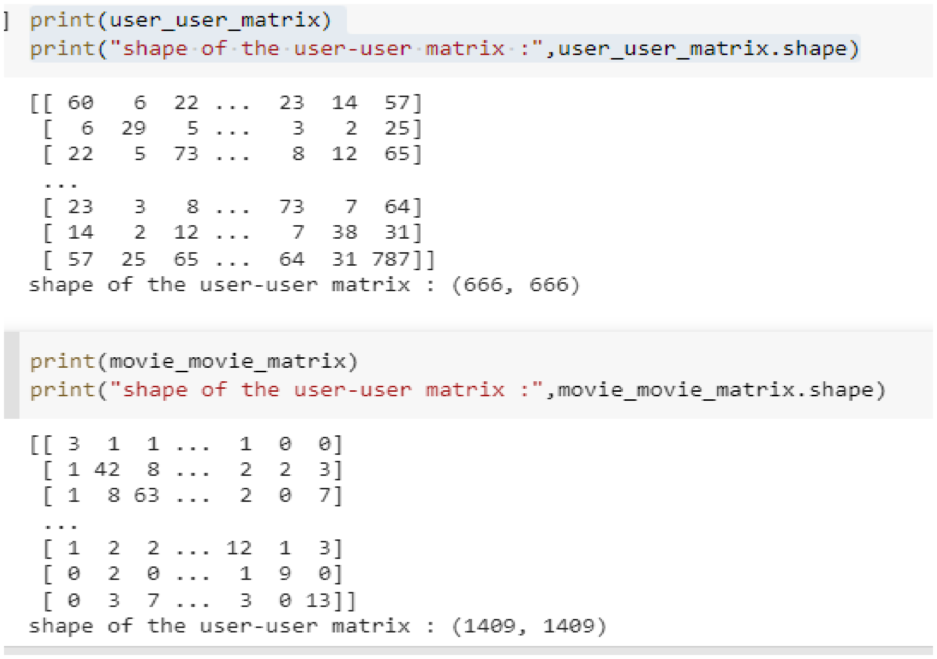
Fig.4.2-Mapping of movie's title with unique id in column of user-movie matrix



```
print(user_movie_matrix)

[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 1 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 1 1 ... 0 0 1]]
```

Fig.4.3-Normalized user-movie matrix

2) Creation of “user-user matrix” and “movie-movie matrix”:

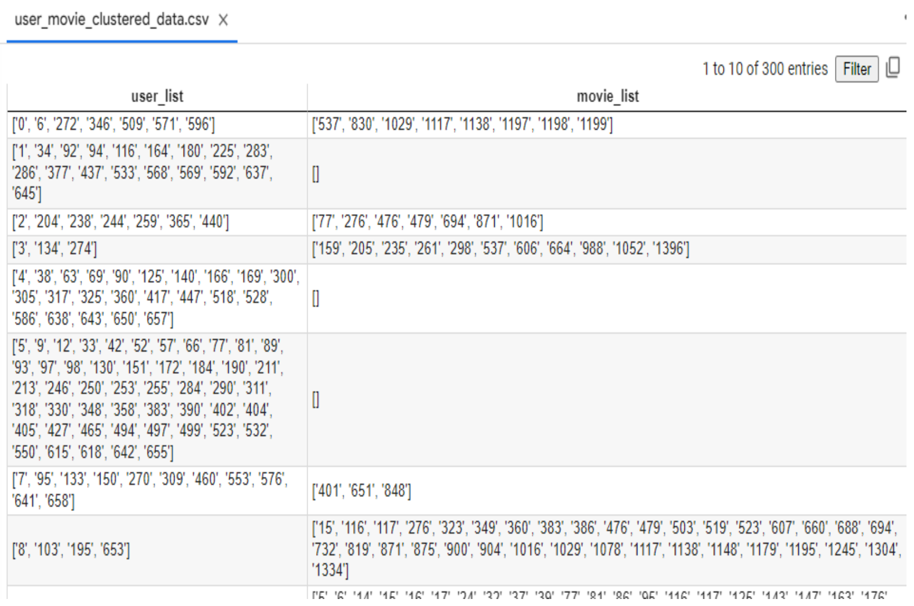


```
]  print(user_user_matrix)
   print("shape of the user-user matrix :",user_user_matrix.shape)

[[ 60   6  22 ...  23  14  57]
 [  6  29   5 ...   3   2  25]
 [ 22   5  73 ...   8  12  65]
 ...
 [ 23   3   8 ...  73   7  64]
 [ 14   2  12 ...   7  38  31]
 [ 57  25  65 ...  64  31 787]]
shape of the user-user matrix : (666, 666)
```

```
print(movie_movie_matrix)
print("shape of the user-user matrix :",movie_movie_matrix.shape)

[[ 3  1  1 ...  1  0  0]
 [ 1 42  8 ...  2  2  3]
 [ 1  8 63 ...  2  0  7]
 ...
 [ 1  2  2 ... 12  1  3]
 [ 0  2  0 ...  1  9  0]
 [ 0  3  7 ...  3  0 13]]
shape of the user-user matrix : (1409, 1409)
```

Fig.4.4-Matrices formed after factorization

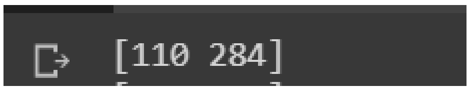
3) Finding clusters from “user user matrix” and “movie movie matrix” and finding corresponding commision movies and users respectively and then storing it into a csv file with header “user_list” and “movie_list”.



user_list	movie_list
['0', '6', '272', '346', '509', '571', '596']	['537', '830', '1029', '1117', '1138', '1197', '1198', '1199']
['1', '34', '92', '94', '116', '164', '180', '225', '283', '286', '377', '437', '533', '568', '569', '592', '637', '645']	[]
['2', '204', '238', '244', '259', '365', '440']	['77', '276', '476', '479', '694', '871', '1016']
['3', '134', '274']	['159', '205', '235', '261', '298', '537', '606', '664', '988', '1052', '1396']
['4', '38', '63', '69', '90', '125', '140', '166', '169', '300', '305', '317', '325', '360', '417', '447', '518', '528', '586', '638', '643', '650', '657']	[]
['5', '9', '12', '33', '42', '52', '57', '66', '77', '81', '89', '93', '97', '98', '130', '151', '172', '184', '190', '211', '213', '246', '250', '253', '255', '284', '290', '311', '318', '330', '348', '358', '383', '390', '402', '404', '405', '427', '465', '494', '497', '499', '523', '532', '550', '615', '618', '642', '655']	[]
['7', '95', '133', '150', '270', '309', '460', '553', '576', '641', '658']	['401', '651', '848']
['8', '103', '195', '653']	['15', '116', '117', '276', '323', '349', '360', '383', '386', '476', '479', '503', '519', '523', '607', '660', '688', '694', '732', '819', '871', '875', '900', '904', '1016', '1029', '1078', '1117', '1138', '1148', '1179', '1195', '1245', '1304', '1334']
	['5', '6', '14', '15', '16', '17', '24', '32', '37', '39', '77', '81', '86', '95', '116', '117', '125', '143', '147', '163', '176',

Fig.4.5-user-movie-clustered.csv file

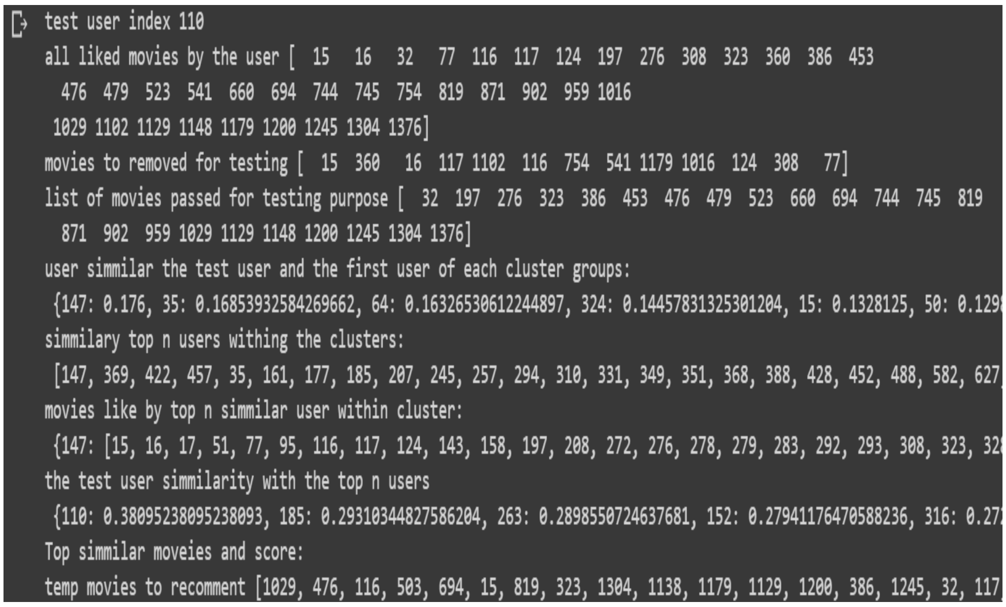
4) Selecting a target user and test inputs(movie ids) for recommendaion:



```
[110 284]
```

Fig.4.6-Target user for testing and recommendations

5) Now selecting the first user element of each user-user cluster and finding its similarity from the test user and sorting the these in descending order and the selecting the top element and further finding the similarity of each user in that cluster with the test user.



```
test user index 110
all liked movies by the user [  15   16   32   77  116  117  124  197  276  308  323  360  386  453
  476  479  523  541  660  694  744  745  754  819  871  902  959 1016
 1029 1102 1129 1148 1179 1200 1245 1304 1376]
movies to removed for testing [  15  360   16  117 1102  116  754  541 1179 1016  124  308   77]
list of movies passed for testing purpose [  32  197  276  323  386  453  476  479  523  660  694  744  745  819
  871  902  959 1029 1129 1148 1200 1245 1304 1376]
user simmilar the test user and the first user of each cluster groups:
 {147: 0.176, 35: 0.16853932584269662, 64: 0.16326530612244897, 324: 0.14457831325301204, 15: 0.1328125, 50: 0.129
simmilary top n users withing the clusters:
 [147, 369, 422, 457, 35, 161, 177, 185, 207, 245, 257, 294, 310, 331, 349, 351, 368, 388, 428, 452, 488, 582, 627
movies like by top n simmilar user within cluster:
 {147: [15, 16, 17, 51, 77, 95, 116, 117, 124, 143, 158, 197, 208, 272, 276, 278, 279, 283, 292, 293, 308, 323, 32
the test user simmilarity with the top n users
 {110: 0.38095238095238093, 185: 0.29310344827586204, 263: 0.2898550724637681, 152: 0.27941176470588236, 316: 0.27
Top simmilar moveies and score:
temp movies to recomment [1029, 476, 116, 503, 694, 15, 819, 323, 1304, 1138, 1179, 1129, 1200, 386, 1245, 32, 117
```

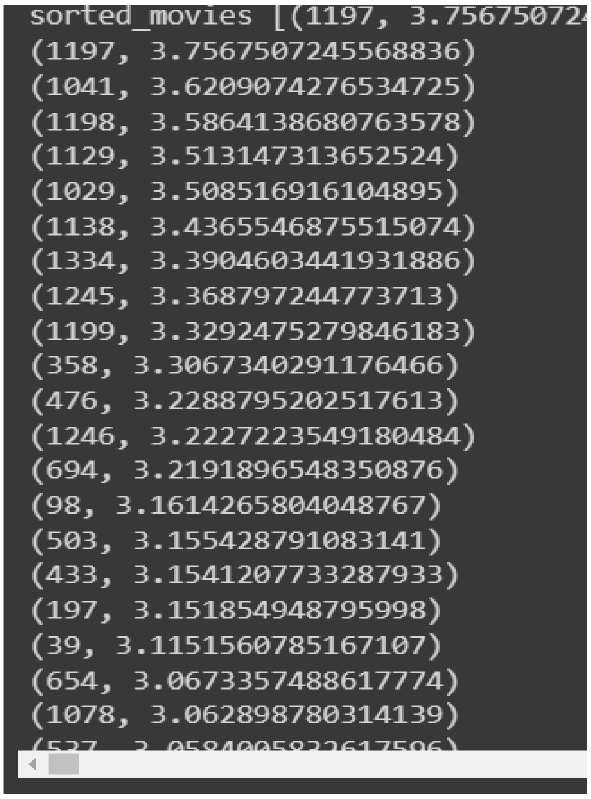
Fig.4.7-Snippet contains essential information regarding test user and test inputs

Here we are first finding the global similarity of the test user with the entire user-user clusters and then from the most similar cluster we further find the local similarity with the user within that particular cluster as we know within a cluster element are similar to each other than outside the cluster elements.

6) Now we have the movie id list with high similarity we chose top n high similarity movie ids and predict the rating for these.
The formula for calculating the predicted rating for a movie, given the ratings of similar users and their similarity scores, is:

predicted_rating=(sum((similarity_score_i)(rating_i)))/(sum(similarity_score_i))*

Where similarity_score_i is the similarity score between the target user and the i-th similar user, and rating_i is the rating given by the i-th similar user for the movie.



```
sorted_movies [(1197, 3.75675072
(1197, 3.7567507245568836)
(1041, 3.6209074276534725)
(1198, 3.5864138680763578)
(1129, 3.513147313652524)
(1029, 3.508516916104895)
(1138, 3.4365546875515074)
(1334, 3.3904603441931886)
(1245, 3.368797244773713)
(1199, 3.3292475279846183)
(358, 3.3067340291176466)
(476, 3.2288795202517613)
(1246, 3.2227223549180484)
(694, 3.2191896548350876)
(98, 3.1614265804048767)
(503, 3.155428791083141)
(433, 3.1541207733287933)
(197, 3.151854948795998)
(39, 3.1151560785167107)
(654, 3.0673357488617774)
(1078, 3.062898780314139)
(537, 3.0584005832617596)
```

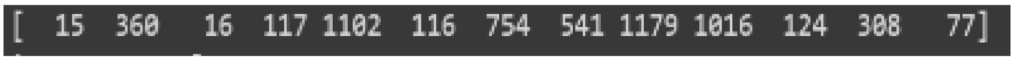
Fig.4.8-A list of recommended movie id with predicted ratings.

This is the list of recommended movie id with their predicted score by the target user.

D. Results:

For the above target user:

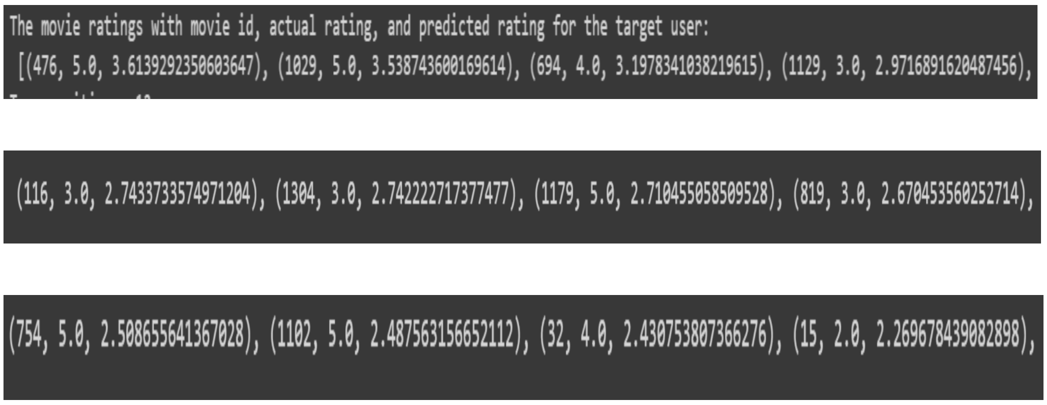
1. The list of movie ids that should be recommended are:



```
[  15  360   16  117 1102  116  754  541 1179 1016  124  308   77]
```

Fig.4.9-list of movies that should be in top recommendations

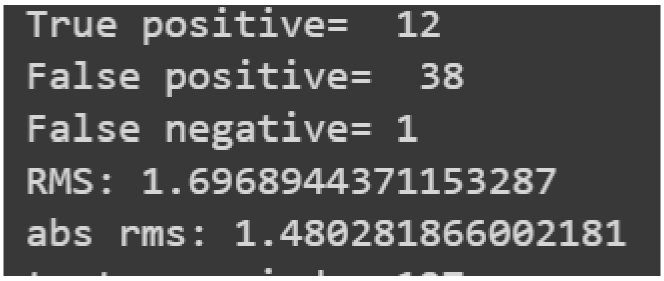
2) The top recommend movie ids are:



```
The movie ratings with movie id, actual rating, and predicted rating for the target user:
 [(476, 5.0, 3.6139292350603647), (1029, 5.0, 3.538743600169614), (694, 4.0, 3.1978341038219615), (1129, 3.0, 2.9716891620487456),

 (116, 3.0, 2.7433733574971204), (1304, 3.0, 2.742222717377477), (1179, 5.0, 2.710455058509528), (819, 3.0, 2.670453560252714),

(754, 5.0, 2.508655641367028), (1102, 5.0, 2.487563156652112), (32, 4.0, 2.430753807366276), (15, 2.0, 2.269678439082898),
```

Fig.4.10 -Top movies with actual and predicted ratings

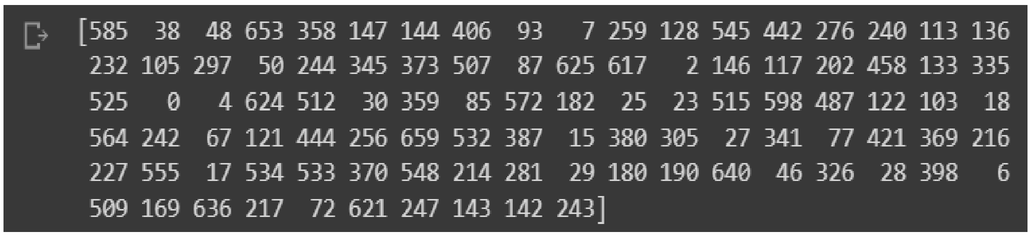
3) Within the top recommended movie ids we have:



```
True positive=  12
False positive=  38
False negative= 1
RMS: 1.6968944371153287
abs rms: 1.480281866002181
```

Fig.4.11-Output represent the evaluation metric for 1 test user

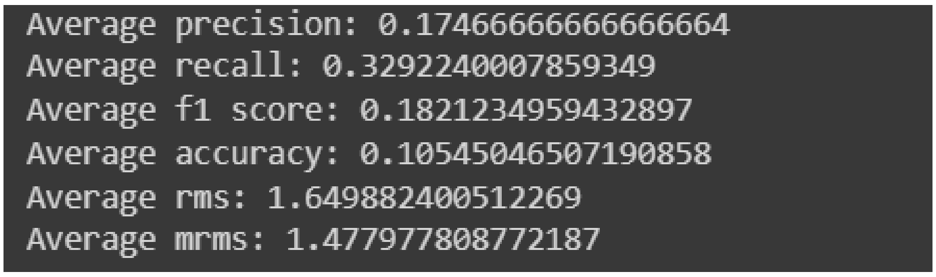
4) Now we run the model on around 100 random test cases of around 700 user, so the random users are:



```
[585  38  48 653 358 147 144 406  93   7 259 128 545 442 276 240 113 136
 232 105 297  50 244 345 373 507  87 625 617   2 146 117 202 458 133 335
 525   0   4 624 512  30 359  85 572 182  25  23 515 598 487 122 103  18
 564 242  67 121 444 256 659 532 387  15 380 305  27 341  77 421 369 216
 227 555  17 534 533 370 548 214 281  29 180 190 640  46 326  28 398   6
 509 169 636 217  72 621 247 143 142 243]
```

Fig.4.12-Random selected test users

5) After having our test user and running these on our system we get the results:



```
Average precision: 0.17466666666666664
Average recall: 0.3292240007859349
Average f1 score: 0.1821234959432897
Average accuracy: 0.10545046507190858
Average rms: 1.649882400512269
Average mrms: 1.477977808772187
```

Fig.4.13- Overall system effectiveness

Here “Average rms” denotes RMSE and “Average mrms” denotes MAE

MAE (Mean Absolute Error) and RMSE (Root Mean Square Error) are two commonly used metrics to evaluate the accuracy of a recommendation system.

Mean Absolute Error (MAE): The average absolute difference between expected and actual ratings is measured by MAE. It's calculated as follows:

$MAE = (1/n) * \sum|\text{predicted rating} - \text{actual rating}|$

where n is the total number of ratings.

MAE is useful when we want to know the average magnitude of the errors in the predicted ratings. The lower the MAE value, the better the accuracy of the recommendation system.

Root Mean Square Error (RMSE): The square root of the average squared discrepancies between expected and actual ratings. It is computed as follows:

*RMSE = sqrt((1/n) * ∑(predicted rating - actual rating)^2)*

RMSE is useful when we want to penalize large errors more than small ones. It is a popular metric because it emphasizes large errors that have a bigger impact on the user experience. The lower the RMSE value, the better the accuracy of the recommendation system.

There are benefits and drawbacks to both MAE and RMSE. The whole range of errors may not be captured by MAE, but it is simple to comprehend and less susceptible to outliers. RMSE, on the other hand, penalises huge mistakes more severely, although it can occasionally be difficult to read and may be susceptible to outliers. The unique application and the objectives of the recommendation system will determine which measure should be used.

CHAPTER-5

CONCLUSION

The biclustering technique to recommendation systems has, in conclusion, yielded encouraging outcomes. In contrast to conventional approaches that only take into account one variable at a time, we were able to provide suggestions that were more accurate by taking into account both user-item preferences concurrently. Additionally, biclustering made it possible to locate significant subgroups in the user-item matrix, leading to a more in-depth comprehension of user preferences and item traits.

We were able to show the efficiency of our recommendation system employing biclustering in a practical environment. The evaluation of these suggestions was done using common metrics like Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). Our algorithm effectively suggested relevant goods to consumers based on their preferences. These findings imply that biclustering has the potential to raise suggestion quality across a range of disciplines.

In the future, there will be a lot of work to be done in the area of recommendation systems, especially in terms of improving the effectiveness of biclustering-based methods.

The applicability of this strategy to more complicated datasets with bigger user and item populations may be explored in further study. However, our project's findings imply that biclustering has a lot of potential as a potent recommendation system tool.

5.2) Future scope:

The concept of biclustering is used in recommendation systems to find groups of users and objects with related preferences. Systems for making recommendations based on biclustering have showed promise in terms of accuracy and scalability. Future research in a number of areas could help biclustering-based recommendation systems perform better. Here are a few potential directions:

a) Temporal biclustering: Most recommendation engines work on the premise that user preferences don't change over time. However, due to changes in their interests, demographics, or other factors, users' preferences may alter over time. To predict changes in user preferences over time and generate more precise suggestions, temporal biclustering may be employed.

b) Hybrid biclustering: Hybrid recommendation systems combine multiple techniques to provide personalized recommendations to users. Biclustering could be combined with other techniques such as collaborative filtering, content-based filtering, and matrix factorization to improve the accuracy and diversity of recommendations.

c) Incorporating context: Contextual data like place, time, weather, and social environment might be added to recommendation algorithms to improve them. Extensions of biclustering-based methods might include context and offer more individualised and pertinent recommendations.

d) Deep biclustering: In a variety of recommendation tasks, deep learning approaches like neural networks have produced encouraging results. In order to enhance the effectiveness of recommendation systems, biclustering may be used in conjunction with deep learning methods.

e) Online learning: The majority of recommendation systems use previous data for offline training. Although user preferences might shift quickly, it might not be practical to constantly retrain the model. As new data becomes available, the model could be updated in real-time using online learning techniques.

Overall, biclustering-based recommendation systems have a lot of potential for future research and development. The above directions can help to improve the accuracy, scalability, and personalized nature of the recommendations provided.

5.3) Applications:

The biclustering approach has been applied to various recommendation system domains, including:

- E-commerce: Customer-specific product suggestions are given through the use of biclustering-based recommendation algorithms on e-commerce platforms. Biclustering can detect client groups with comparable tastes and recommend goods that are well-liked by these groups.
- Music recommendation: Users and songs may be grouped together using biclustering based on auditory characteristics including pace, melody, and timbre. As a result, users may receive more personalised, accurate, and varied music choices.
- Health care: Biclustering may be used to analyse patient subgroups with related medical diagnoses, demographics, and genetic profiles using electronic health record (EHR) data. Better health outcomes and more individualised treatment suggestions may result from this.
- Social media: Systems for recommending friends, groups, and material have been implemented in social media platforms using biclustering. Biclustering may find user groups with shared interests and recommend material that appeals to these groups.
- Movie recommendation: Biclustering may be used to find user groups with comparable movie tastes and recommend films that are well-liked within these groups. As a result, users may receive more individualised movie suggestions that are accurate and varied.
- Travel and tourism: In the travel and tourism sector, biclustering may be used to group people according to their travel tastes and recommend locations and travel deals that are well-liked within these clusters. This may result in more individualised and pertinent travel advice.

- Education: Biclustering may be used in education to group students according to their academic standing, preferred learning method, and hobbies and then provide activities and resources that are specifically catered to their requirements. This may enhance academic performance and student involvement.

- Finance: Biclustering can be used in the financial industry to locate groups of clients with comparable financial profiles and provide tailored investment plans and financial products that are appropriate for their requirements. Better investment choices and increased consumer loyalty may result from this.

Overall, biclustering-based recommendation systems can be applied to a wide range of domains where personalized recommendations are required, and users and items have multiple attributes.

REFERENCES

[1] F. Alqadah, C. K. Reddy, J. Hu, and H. F. Alqadah, "Biclustering neighborhood-based collaborative filtering method for top-n recommender systems," Knowledge and Information Systems, vol. 44, no. 2, pp. 475-491, 2015.

[2] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to Bregman co-clustering and matrix approximation," in KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 509-514.

[3] T. George and S. Merugu, "A Scalable Collaborative Filtering Framework Based on CoClustering," in Proceedings of the 5th IEEE International Conference on Data Mining (ICDM 2005), 27-30 November 2005, Houston, Texas, USA, pp. 625-628, IEEE Computer Society, 2005.

[4] T. M. Murali and S. Kasif, "Extracting Conserved Gene Expression Motifs from Gene Expression Data," in Proceedings of the 8th Pacific Symposium on Biocomputing, PSB 2003, Lihue, Hawaii, USA, January 3-7, 2003, pp. 77-88, 2003.

[5] A. Prelic, S. Bleuler, P. Zimmermann, A. Wille, P. B¨uhlmann, W. Gruissem, L. Hennig, L. Thiele, and E. Zitzler, "A systematic comparison and evaluation of biclustering methods for gene expression data," Bioinformatics, vol. 22, no. 9, pp. 1122-1129, 2006.

[6] M. Singh and M. Mehrotra, "Impact of biclustering on the performance of Biclustering based Collaborative Filtering," Expert Syst. Appl., vol. 113, pp. 443-456, 2018.

[7] P. Symeonidis, A. Nanopoulos, A. N. Papadopoulos, and Y. Manolopoulos, "Nearest-biclusters collaborative filtering based on constant and coherent values," Inf. Retr., vol. 11, no. 1, pp. 51-75, 2008.

[8] Florestan de Moor. A Biclustering Approach to Recommender Systems. Machine Learning [cs.LG]. 2019. ffhal-02369708.

[9] O. Maâtouk, W. Ayadi, H. Bouziri and B. Duval, "Evolutionary Local Search Algorithm for the biclustering of gene expression data based on biological knowledge," Applied Soft Computing, vol. 104, pp. 107177, 2021, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2021.107177.

[10] L. Tondji, "Web Recommender System for Job Seeking and Recruiting," 2018, doi: 10.13140/RG.2.2.26177.61286.

[11] Bai, Xiaomei & Wang, Mengyang & Lee, Ivan & Yang, Zhuo & Kong, Xiangjie & Xia, Feng. (2019). Scientific Paper Recommendation: A Survey. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2890388.

[12] I. Khater, I. Nabi and G. Hamarneh, "A Review of Super-Resolution Single-Molecule Localization Microscopy Cluster Analysis and Quantification Methods," Patterns, vol. 1, pp. 100038, 2020, doi: 10.1016/j.patter.2020.100038.

[13] R. Alves, D. S. Rodríguez Baena and J. Aguilar-Ruiz, "Gene association analysis: A survey of frequent pattern mining from gene expression data," Briefings in Bioinformatics, vol. 11, pp. 210-224, Oct. 2009, doi: 10.1093/bib/bbp042.

[14] F. Almatrooshi, I. Akour, S. Alhammadi, K. Shaalan and S. Salloum, "A Recommendation System for Diabetes Detection and Treatment," in 2020 5th International Conference on Computing, Communications and Control Technology (I4CT), 2020, pp. not specified, doi: 10.1109/CCCI49893.2020.925667

bicluster

ORIGINALITY REPORT

9%	5%	4%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.aporia.com Internet Source	1%
2	Submitted to University of Greenwich Student Paper	1%
3	waset.org Internet Source	1%
4	Submitted to Middle East College of Information Technology Student Paper	1%
5	Faris Alqadah, Chandan K. Reddy, Junling Hu, Hatim F. Alqadah. "Biclustering neighborhood-based collaborative filtering method for top-n recommender systems", Knowledge and Information Systems, 2014 Publication	<1%
6	Submitted to Liverpool John Moores University Student Paper	<1%
7	Submitted to Nanyang Technological University Student Paper	<1%