Revue d'Intelligence Artificielle
Vol. 38, No. 1, February, 2024, pp. 159-173
Journal homepage: http://iieta.org/journals/ria

Towards Optimizing Hybrid Movie Recommender Systems

Hassan A. Khalil

Department of Mathematics, Faculty of Science, Zagazig University, Zagazig 44519, Egypt

Corresponding Author Email: h.a.khalil@zu.edu.eg

https://doi.org/10.18280/ria.380116

Received: 26 September 2023
Revised: 2 December 2023
Accepted: 8 January 2024
Available online: 29 February 2024

Keywords:
content-based recommender systems, collaborative filtering recommender systems, hybrid recommender systems, root mean square error, matrix factorization, resilient distributed dataset, spark

ABSTRACT

Research in movie recommendation systems addresses several specific challenges to enhance the accuracy, relevance, and user satisfaction of the recommendations. Some of the key challenges include sparsity of data and cold start for new movies. Integrating collaborative filtering and content-based filtering in a seamless and effective manner also poses another key challenge. This article explores strategies for optimizing hybrid recommendation systems to leverage the strengths of each approach. By addressing these three key challenges, we aim to advance the state of movie recommendation systems, providing more effective and user-centric solutions that cater to the diverse preferences and needs of movie enthusiasts. Similarity-based models (memory-based) combined with matrix factorization-based models (model-based) were used to generate an optimal hybrid movie recommendation model. We evaluated the models using a large dataset of movies and we showed that the proposed model is both efficient and scalable.

1. INTRODUCTION

In the digital age, where information is abundant and easily accessible, individuals often face a challenge commonly known as "information overload." This phenomenon occurs when the volume of available information surpasses an individual's capacity to process it effectively.

As a result, users may find it overwhelming to navigate through the vast sea of data to discover content that aligns with their preferences and interests. Recommendation systems play a crucial role in addressing this challenge by leveraging algorithms and data analysis to provide personalized suggestions to users [1, 2]. These systems are designed to understand user behavior, preferences, and patterns based on their interactions with digital platforms. Whether it's streaming services, e-commerce websites, social media platforms, or news outlets, recommendation systems help users discover relevant content amidst the abundance of options.

In essence, recommendation systems serve as navigational aids in the vast digital landscape, offering users a curated and personalized pathway through the abundance of available data. As data overload continues to be a defining characteristic of the digital age, the role of recommendation systems becomes increasingly crucial in providing users with meaningful, relevant, and personalized experiences across various online domains.

Many different fields have studied recommender systems: e-commerce [3-6], web search [7-9], information retrieval [10, 11], job boards [12-19], movie ratings [20-23], etc.

In the majority of movie recommender systems, a movie's usefulness is often stated by a score that reflects how a particular user felt about a movie in question [24-27]. Predicting a score for movies that are not yet reviewed by users can be one of the greatest challenges in building movie recommender system.

Due to the large number of movies and users in the system, it can be difficult for each user to see all movies and to rate every one individually. The user can be guided to the movies with the highest estimated ratings when it is possible to estimate scores for movies that haven't yet been evaluated.

The three most popular approaches are content-based (CB) [28-31], collaborative filtering (CF) [32-36], and hybrid (which combine the two prior approaches) [37-41].

Optimizing movie recommender systems offers tangible benefits to users, the industry, and the academic community. It enhances user experiences, contributes to industry growth, fosters academic collaboration and innovation, and addresses ethical considerations, ultimately shaping the landscape of personalized content recommendations.

For streaming platforms and entertainment services, optimized recommender systems contribute to increased user engagement and retention. By offering compelling movie recommendations, these platforms can keep users actively involved, leading to longer subscription periods and higher customer loyalty.

From industry players' side, they can leverage optimized recommender systems to enhance personalized advertising. By understanding user preferences, the system can deliver targeted ads, leading to higher click-through rates and more effective advertising campaigns.

From users' side, optimized hybrid recommender systems can encourage users to explore new movie genres they might not have considered. This not only broadens users' cinematic horizons but also supports a diverse and inclusive entertainment ecosystem.

Research in optimizing hybrid movie recommender systems contributes also to the academic community by advancing knowledge in the fields of machine learning (ML), data science, and recommendation systems. It provides insights into novel algorithms, optimization techniques, and evaluation methodologies, contributing to the academic discourse and shaping the future of recommendation systems.

The scope of this paper extends beyond movie recommendation systems to encompass a broader examination of recommender systems in various domains within the digital landscape. While movie recommendation systems may be used as illustrative examples, the focus is not limited to this specific application.

The main contributions of this article are:

- Exploring strategies for optimizing hybrid recommendation systems to leverage the strengths of each approach used in the combination.
- Providing more effective and user-centric solutions that cater to the diverse preferences and needs of movie enthusiasts.
- Optimizing recommender models based on similarities (memory-based) and models based on matrix factorization (model-based).
- Modeling an efficient and scalable hybrid movie recommendation system.

The remainder of this paper consists of seven sections. Recommender systems methods are highlighted in Section 2. Optimization of recommender systems are given in section 3. Experiments are detailed in section 4. Solution establishment is presented in section 5. Section 6 presents discussion of the research work. Finally, section 7 concludes the article and highlights the research perspectives.

2. METHODS OF RECOMMENDER SYSTEM: A LITERATURE REVIEW

Recommender systems are classified according to the used approach to estimate missing scores.

CB approaches: The user will be given r Recommender systems are classified according to the used approach to estimate missing scores.

- CB approaches: they recommend items based on their attributes and features, matching them with user preferences. It relies on item profiles and user profiles constructed from explicit or implicit feedback.

Spotify's recommendation system employs content-based filtering by analyzing the audio features of songs (such as tempo, key, and genre) and matching them to users' listening history to suggest music that aligns with their preferences.

The user will be given recommendations for movies that are comparable to those he has previously preferred in terms of the degree of similarity between the movies [28-31]. Consider example 1 in Table 1, where users rate movies. It shows the known scores for each user (in row) and for each movie (in column). In the example presented in Table 1, the movies TAG and Insidious: The Red Door were rated almost the same by John. We also observe that the characteristics of the movies TAG and Elemental are very close. The system therefore recommends the movie Elemental to John, assigning it the best score it has given to the movies TAG and Insidious: The Red Door, i.e. 8.

- Collaborative filtering (CF) approaches: it relies on user-item interaction data to make recommendations. It can be user-based or item-based, and it identifies patterns and similarities between users or items to predict preferences.

Netflix's recommendation system utilizes collaborative filtering to suggest movies and TV shows based on user viewing history and preferences. The system analyzes user behavior and similarities with other users to make personalized recommendations.

Movies that previous users with comparable tastes and preferences have liked will be suggested to the user [32-36]. Let's go back to example 1 of Table 1. John and Miguel both like TAG and Insidious: The Red Door and didn't like Joy Ride; their similar opinions on these movies suggest that in general Miguel and John are of the same opinion. So Miraculous is a good recommendation for John, since Miguel loves her. The system thus assigns to the movie Miraculous, for John, the score that Miguel gave to this movie, 9.

- Hybrid strategies: a fusion of the two methods mentioned above [37-41]. They combine multiple recommendation approaches to benefit from their strengths and mitigate their weaknesses. This can involve integrating collaborative filtering, content-based filtering, or other techniques.

YouTube employs a hybrid recommender system that incorporates collaborative filtering, content-based filtering, and deep learning models. This enables the platform to consider both user behavior and video content features for more accurate and diverse recommendations.

Table 1. Sample data (example 1)

$u(c, i)$	Miraculous	TAG	Elemental	Joy Ride	Insidious: The Red Door
John	-	8	-	2	7
Miguel	9	8	-	3	6
Sarra	3	5	-	5	-
Lama	5	3	-	3	3
John	-	8	-	2	7

2.1 CB recommender systems

The objective of CB methods is to identify which catalog movies most closely fit the user's tastes [28-31]. A huge user base or extensive system usage history are not necessary for this method. Figure 1 illustrates this process.

An explicit list of each movie's attributes is the simplest approach to define a catalog of movies. One might use the genre, authors' names, publisher, or any other details about a book. A list of interests based on the same attributes is used to express the user profile.

There are several ways to gauge how well the elements' qualities match the user's profile:

- The Dice index or other similarity measures [42].
- The TF-IDF (Term Frequency-Inverse Document Frequency) [43].

- Techniques based on the similarity of vector spaces (Bayesian approaches [44], decision trees [45], etc.) coupled with statistical techniques.

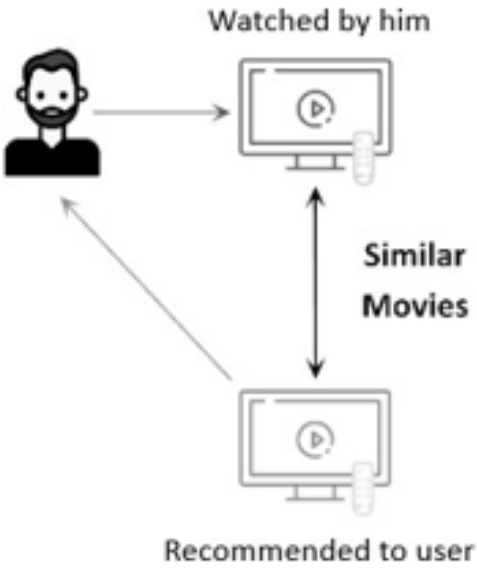

Figure 1. CB movie recommender systems

CB have the following advantages:

- They recommend similar movies that others have liked.
- In order to provide each user with the most appropriate recommendations, they take into account their profile.
- Since lists of keywords are frequently used, matching user preferences and movie characteristics is effective for many forms of data (textual, quantitative, etc.).
- Users' personal information is useless.
- When a new movie is added to the catalog, there is no cold start issue or low density since the preferences of the users are considered [46].

However, such approaches also have drawbacks:

- It is impossible to discriminate between movies represented by the same collection of keywords.
- Users who have viewed a lot of movies can be problematic because their profiles contain too much information that doesn't fit the qualities of the movies.
- There is no history when a new user first uses the system.
- User profiles are still challenging to create, and it's also important to consider how a user's interests have changed over time.

2.2 CF recommender systems

Systems built on the CF model generate recommendations by comparing a user's preferences to those of other users. The substance of the movies to be recommended is not examined or attempted to be understood by such systems. The technique involves gathering reviews from numerous users to automatically forecast a user's preferences. The essential premise of this strategy is that people who previously enjoyed a specific movie would probably continue to enjoy it or something quite similar in the future [32-36]. Figure 2 illustrates this process.

The idea of collaborative techniques is to foresee users perceptions of the various components. The suggestion is based on the user's prior preferences and viewpoints as well as on a user similarity metric. The key actions in this strategy are:

- Many user preferences are saved.
- The user seeking the recommendation belongs to a subgroup of people with similar preferences.
- The preferences for this group are averaged out; movies are suggested to users who request recommendations using the resultant preference feature.

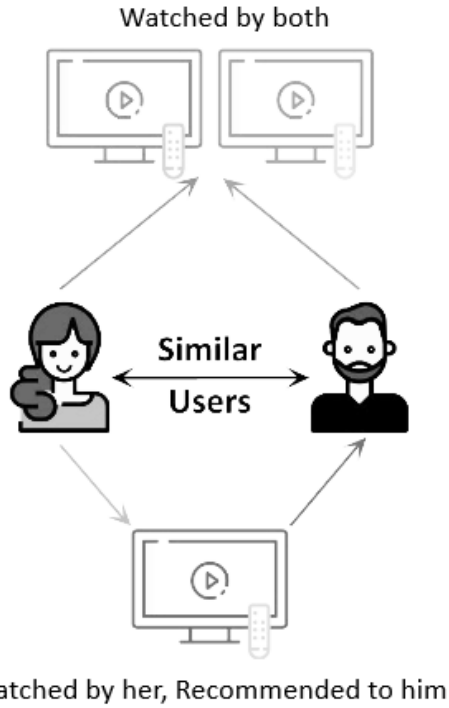

Figure 2. CF movie recommender systems

Three different techniques of defining similarity can be identified:

- Item-to-Item approaches are based on how similar two movies are to one another. It should be noted that this strategy works for a very broad range of users or elements.
- User-to-User approaches are predicated on user similarities. They are not appropriate for a large number of users.

We have already used the User-to-User approach in Example 1: Miguel and John have similar opinions; Miguel also likes Miraculous, so it is a good recommendation for John. Let's now use the Item-to-Item approach on this same example. John and Miguel love TAG and Insidious: The Red Door. This suggests that, in general, people who like TAG will also like Insidious: The Red Door, so Insidious: The Red Door might be recommended to Sarra (who likes TAG).

Due to their diversity, CF recommender systems are consequently based on a variety of methodologies, including:

- User-to-user methods that consider user similarity or neighborhood selection (e.g., algorithms based on neighborhood search).
- The cosine similarity metric, among other things, for item-to-item techniques.

- Techniques for predicting scores for other approaches (Principal Component Analysis [47], matrix factorization [48], latent semantic analysis [49], association rules [50], Bayesian approaches [51], etc.).

The benefits of CF recommender systems are as follows:

- Applying user ratings to assess the utility of products.
- Locating individuals or groups of individuals whose interests coincide with the current user.

However, such systems also have drawbacks:

- It's challenging to locate users or groups of users who are similar.
- The $User\ x\ Item$ matrix's low density hinders the recommendation mechanism.
- There is also the cold-start issue: a new user's preferences are unknown when they use the system, and no one is given credit for him when a new movie is uploaded to the catalog.
- The computation expands linearly in systems with many users and things; hence, suitable methods are required.
- non-diversity: If a user enjoys one Antonio Banderas movie, it is not helpful to suggest all of his movies to them.

Amazon.com customizes its online store for each consumer using recommendation algorithms. The store drastically alters depending on the preferences of the customer, displaying baby toys to new parents or programming movies to a software developer. Collaborative filtering on movies, also known as item-to-item filtering, is the basis of the algorithm used by Amazon.com. With this algorithm, the number of movies in the catalog is taken into account as well as the number of customers.

2.3 Hybrid recommender systems

A hybrid recommender system relies on the logic of multiple types of recommendation systems or uses its constituent parts [37-40]. By merging CF and CB approaches, such a system can make use of both external knowledge and movie attributes [52]. Being a blend of methodologies, it highlights the benefits of such approaches while minimizing their drawbacks (see Figure 3).

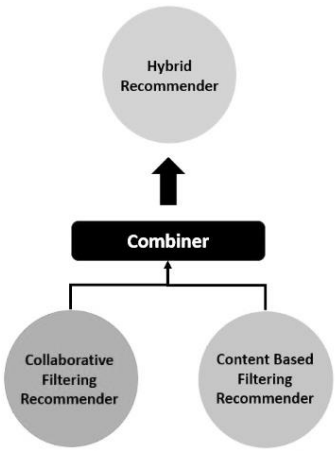

Figure 3. Hybrid recommender system

The term hybrid refers to the historical development of recommender systems, which involved the first exploitation of specific information sources to produce well-established approaches that were later integrated. The goal is to employ as many different sources of knowledge as you can, selecting the ones that are most pertinent to the work at hand.

Monolithic refers to a hybridization design where a single algorithm incorporates elements of various recommendation systems [53-55]. As shown in Figure 4, multiple recommender systems contribute to this because the hybrid approach uses additional input data that is specific to another recommender algorithm or else the input data is upgraded by one technique and used by another.

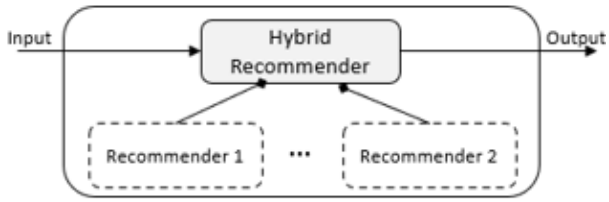

Figure 4. Monolithic hybrids

This includes, for instance, a CB recommendation system that also uses community data to identify movie similarities.

The other two hybrid strategies demand combining at least two different recommendation implementations [55, 56]. As shown in Figure 5, parallel hybrid recommender systems function independently of one another and generate unique suggestion lists based on their input data. Their results are merged into a final set of suggestions in a subsequent hybridization process.

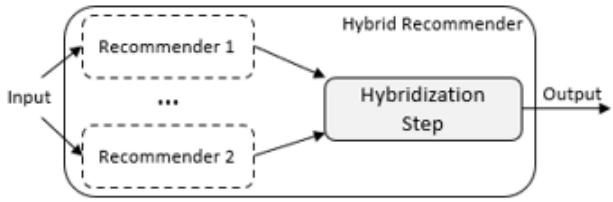

Figure 5. Parallelized hybrids

The output of one recommender system becomes a component of the input data for the following system when many recommender systems are connected in a pipe architecture, as shown in Figure 6 [55, 57].

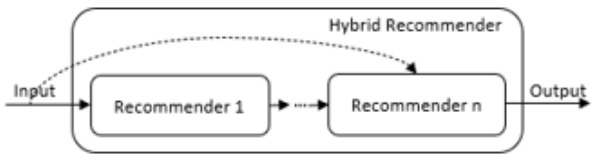

Figure 6. Pipelined hybrids

Each day, hundreds of millions of tailored recommendations are made to customers by Netflix's recommendation engine, named Cinematch, based on an analysis of overall movie ratings [58].

The Cinematch recommender system automatically analyzes cumulative movie scores on a weekly basis using a variant of Pearson's Correlation Coefficient [59] with all other movies in order to produce a list of films that are likely to be enjoyed. As the user submits their ratings, the online, real-time

component of the system performs a multivariate regression based on these correlations to produce an individual, personalized prediction for each recommendable movie based on these scores. In lack of a customized recommendation, the overall average of the movie's scores is used.

3. OPTIMIZING RECOMMENDER SYSTEMS

3.1 Movies dataset

GroupLens Research has collected and made available data from the MovieLens website http://movielens.org. The first dataset, named ratings.csv, of 100004 lines, contains the ratings that users have given to movies (see Table 2):

Table 2. Rating sample data

	UserID	MovieID	Rating	Timestamp
0	**1**	31	2.5	1260759144
1	**1**	1029	3.0	1260759179
2	**1**	1029	3.0	1260759182
3	**1**	1129	2.0	1260759185

The second dataset, named movies.csv, contains movie information, including Title and Genres concatenated in a string (see Table 3).

Table 3. Movies sample data (Bennett & Lanning, 2007)

	MovieID	Title	Genres
0	**1**	Toy Stoty (1995)	Adventure |Animation| Children| Comedy| Fantasy
1	**2**	Jumanji (1995)	Adventure| Children| Fantasy
2	**3**	Grumpier Old Men (1995)	Comedy| Romance
3	**4**	Waiting to Exhale (1995)	Comedy| Drama| Romance

We will then present the IMDbPY, a Python package for retrieving and managing the data of the IMDb (Internet Movie Database) package. It contains more characteristics relating to movies. This allowed us to build a new dataset containing new variables that best characterize the movies.

3.2 Model overview

A recommendation system is a kind of specific form of filtering information aimed at presenting the elements likely to interest the user. Setting up a recommender generally requires 3 steps:

- **Data collection**: in the basic case, these are either the characteristics of movies or users, or the rating submitted by the user to a which is generally presented in the form of a rating out of 5 stars or binary events such as like and follow.
- **Model building**: In general, it is a matrix that is often presented in the form of a table which crosses the users and the movies according to the main event.
- **Extract recommendations**: recommendations are extracted from models and then found for a user, other users, or the movies that are most comparable to them by applying filters or processing.

3.3 Optimizing CF movie recommender system

Ensuring diversity and novelty in recommendations is crucial for enhancing user satisfaction in a recommender system. Users often appreciate a system that not only suggests items they are likely to enjoy but also introduces them to new and diverse content.

In optimizing CB recommendation systems, the features used to represent items can be diversified. For example, if recommending movies, features could include genre, director, actors, and more. This allows the system to recommend items that differ in various aspects, promoting diversity.

CF has been modified to ensure that user profiles are diverse. It has been achieved by incorporating diversity measures in the recommendation algorithm.

Optimizing hybrid models that combine both CB and CF techniques leverage the strengths of each optimizing approach. This leads to recommendations that are not only accurate but also diverse, as they consider both item features and user preferences.

We will also incorporate exploration-exploitation strategies to balance between recommending items that are similar to the user's known preferences (exploitation) and suggesting movies that are new or less explored (exploration). This helps in introducing novelty while still providing familiar recommendations.

By employing these strategies, recommender systems strike a balance between accuracy and diversity, providing users with recommendations that are not only personalized but also introduce them to novel and diverse items. This, in turn, contributes to a more satisfying user experience.

3.3.1 Memory-based CF

The two primary categories of memory-based CF techniques are item-item filtering and user-item filtering [60], [61]. A user-item filter will look up a certain user's ratings to locate people who are similar to that person and then suggest movies that those similar users enjoyed (Users who are similar to you, also liked ...). Item-item filtering, in contrast, takes a movie, identifies users who enjoy it, and identifies more movies that those users or similar users also enjoy. It accepts something and outputs something else, like recommendations (Users who liked this also liked…).

In both cases, and as a first step, we create the User Item matrix from all the data where we will represent the crossing of users and movies through the ratings. Since in such a case we have to go through cross-validation, we will obtain two User Item matrices: Train and Test. The first will contain 75% of the ratings, while the second will have 25% of the latter. Figure 7 shows an example of User × Item matrix.

Figure 7. Sample of User × Item matrix

We must determine similarity and build a similarity matrix after building the User Item matrix.

In the item-item scenario, the similarity values between two movies i and j are calculated by looking at all the users who have rated the two movies (see Figure 8).

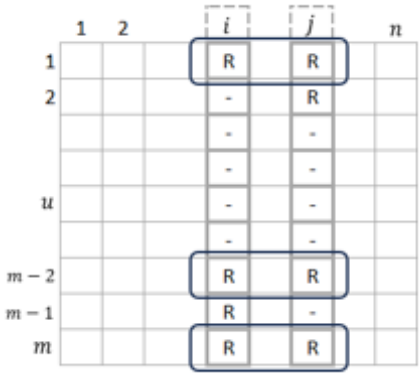

Figure 8. Item × Item matrix

By looking at all the movies that the two users score, the user-item scenario compares the similarity values between two users i and j (see Figure 9).

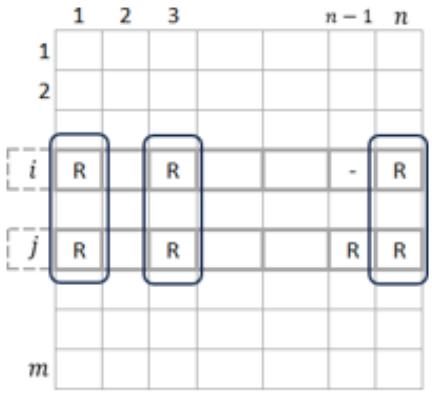

Figure 9. User × Item matrix

To do this, we used the following two metrics on the set of training data (Train) for the two user-item and item-item cases:

1) Cosine similarity: A measurement that determines the cosine of the angle between two vectors [62, 63]. Since we take into account both the angle between the movies as well as the relevance of each word count (TF-IDF), this metric can be thought of as a comparison between things in a normalized space (see Figure 10). We must solve Eq. (1).

$$\vec{a} \cdot \vec{b} = \| \vec{a} \| \| \vec{b} \| \cos\theta \tag{1}$$

$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\| \vec{a} \| \| \vec{b} \|} \tag{2}$$

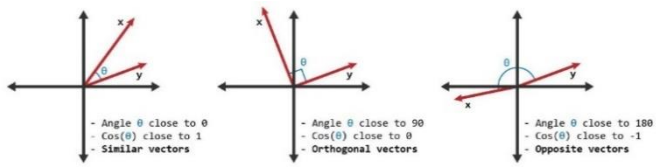

Figure 10. Cosine similarity

2) City Block distance: This is a measure of distance that is calculated as the average difference between dimensions. In most cases, this distance produces results close to those obtained by the simple Euclidean distance [64]. However, with this measure, the effect of a single large difference (outliers) is mitigated (because it is not squared).

The City Block distance between vectors u and v is given in the Eq. (3).

$$CityBlock(u, v) = \sum_{i}^{N} |u_i - v_i| \tag{3}$$

3.3.2 Model-based CF

The second subtype of CF, named model-based CF, will now be used. It entails using the factorization matrix, which is an unsupervised learning technique for dimensionality reduction and decomposition of hidden variables [65]. By multiplying the user and item latent variable matrices, the factorization matrix aims to predict unknown ratings by learning hidden user preferences and hidden object features from known ratings in our dataset. The implementation of recommender systems uses several dimensionality reduction approaches. In conducting our study, we employed:

1) Singular Value Decomposition (SVD): this method entails lowering the dimensionality of the *User* × *Item* matrix that was previously computed [66].

Let *R* be the *User* × *Item* matrix of size *m* × *n* (*m*: number of users, *n*: number of movies), and *k*: the dimension of the

latent character space. The general SVD equation is given by Eq. (4).

$$R = USV^T \quad (4)$$

where:

- U $(m \times k)$ is the array of latent characters for users.
- V $(n \times k)$ is the array of latent characters for movies.

The diagonal matrix of size $k \times k$ with real positive values. By multiplying the three matrices, we can make the prediction.

2) Stochastic Gradient Descent (SGD): we wish to estimate two matrices, P $(m \times k)$ of latent characters for users and Q $(n \times k)$ of latent characters for movies [67]. The unknown ratings can then be predicted by multiplying the P and Q matrices after the estimation of P and Q.

To update P and Q, we can use the SGD where we iterate each observation in the train to update P and Q as we go (see Eq. (5) and Eq. (6)).

To update P and Q, we can use the SGD where we iterate each observation in the train to update P and Q as we go (see Eq. (5) and Eq. (6)).

$$\vec{a} \cdot \vec{b} = \| \vec{a} \| \| \vec{b} \| cos\theta \quad (5)$$

$$\vec{a} \cdot \vec{b} = \| \vec{a} \| \| \vec{b} \| cos\theta \quad (6)$$

where:

-γ is the learning speed.

-λ is the regularization term.

-e is the error which is the difference between the actual rating and the predicted rating.

The SGD model is based on several steps defined beforehand.

3) Alternating Least Squares (ALS): our goal with ALS is to estimate P and Q. Then, by multiplying the transpose of the P matrix by the Q matrix, we may estimate the unknown ratings after estimating P and Q [68].

Table 4. presents the strengths and weaknesses of memory-based and model-based approaches

	Strengths	Weaknesses
Memory-Based Approaches	- **Simplicity and Intuitiveness**: Memory-based methods are often straightforward to understand and implement. They rely on the similarity between instances, making them intuitive for many applications. - **No Training Phase:** Memory-based systems do not require a separate training phase. They immediately adapt to new data, making them suitable for dynamic environments where the data distribution can change over time. - **Transparency**: The reasoning behind the recommendations is transparent since it's based on the similarity of instances. Users can easily understand why a certain recommendation was made. - **Handling Cold Start Problem:** Memory-based methods can handle the cold start problem well, as they don't rely on pre-existing models. They can provide recommendations for new items based on their features and similarity to existing items	- **Scalability**: Memory-based approaches may struggle with scalability as the size of the dataset increases. The computation cost of finding similarities among all data points can become prohibitive. - **Sparsity**: In sparse datasets, where users' interactions with items are limited, finding meaningful similarities becomes challenging. This can result in poor recommendations, especially for less popular items. - **Lack of Personalization**: Memory-based systems may not capture complex patterns or individual preferences well, leading to less personalized recommendations compared to model-based approaches
Model-Based Approaches	- **Scalability**: Model-based approaches, particularly those using machine learning models, can handle large datasets more efficiently. They often scale better as the data size grows. - **Personalization**: Model-based methods can capture complex patterns and individual preferences better than memory-based methods. This results in more personalized recommendations, especially in scenarios with diverse user preferences. - **Feature Learning**: Model-based approaches can automatically learn relevant features and representations from the data, allowing them to adapt to the underlying patterns without manual feature engineering. - **Handling Sparsity**: Model-based methods can perform better in sparse datasets by learning latent factors and relationships between users and items, effectively dealing with the sparsity problem	- **Complexity**: Model-based methods are often more complex to implement and may require a dedicated training phase. The need for optimization and hyperparameter tuning can add to the complexity. - **Cold Start Problem**: Model-based systems may struggle with the cold start problem, especially for new items or users, as they rely on historical data for training - **Lack of Transparency**: The inner workings of complex models may be less transparent compared to memory-based approaches. This lack of transparency can be a drawback in applications where interpretability is crucial

According to Table 4, we adopt the following optimization scenario:

1) Choose Memory-Based Approaches When:

- Transparency and simplicity are priorities.
- The dataset is not too large, and scalability is not a primary concern.
- Cold start and adaptability to new data are critical.

2) Choose Model-Based Approaches When:

- Scalability and handling large datasets are essential
- Personalization and capturing complex patterns are crucial.
- A training phase can be incorporated, and transparency is not a top priority.

Ultimately, the choice between memory-based and model-based approaches depends on the specific requirements and constraints of the recommendation system and the characteristics of the data it will operate on.

3.3.3 Optimizing CB recommender systems

Object recommendation or CB is different from CF because it is not based on past user behavior. It involves recommending objects based on the intrinsic qualities and properties of the object itself and correlating them with the preferences and interests of the user. For our case, we will only be interested in the correlation of an object with another object.

The movies.csv dataset admits 3 variables MovieID which represents the identifier of the movie, Title the name of the movie and Genres which makes it possible to categorize the movies according to their themes knowing that a movie can have multiple genders.

To start, we treated the similarity between the movies based only on the Genres variable. This did not seem to us sufficient in terms of the degree of similarity, so we thought of using the IMDbPY python package to have new variables that best characterize the movies in our dataset. The new variables added are: Genre, Year, Kind, Director, Cast, Writer, Rating, Runtimes, Countries, Languages, and Production companies.

Figure 11 shows how to import movie data using IMDbPY https://github.com/MaximShidlovski23/imdbpy.

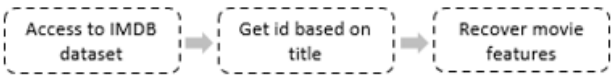

Figure 11. Importing movie data using IMDbPY

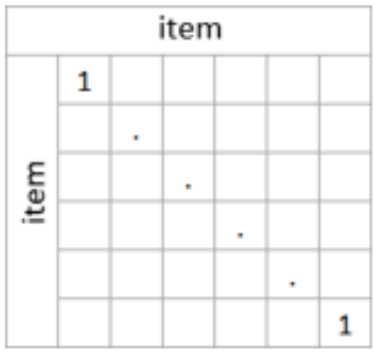

Figure 12. Square matrix

Now that we have the characteristics of all the movies from their titles and their IMDB identifiers, we can create our new dataset with all the movies and their characteristics.

Now, the goal is to build a square matrix (l x l: where l is the number of movies) that contains the distance between each pair of movies in order to determine which things are closest to one another and hence share the most similarities (see Figure 12).

3.4 Optimizing hybrid recommender systems

After presenting CF and CB, the idea behind this part is to create a model that combines the results of the two previous methods. In this part we tried to combine between the similarity scores of the movies (compared to a certain movie) obtained by the two methods:

CF and CB.

4. EXPERIMENTS

4.1 Optimizing CF recommender systems

One must use an evaluation metric to gauge the precision of the anticipated estimations in order to evaluate the similarity. The Root Mean Square Error (RMSE), one of the two primary performance measures for a regression model, is one of the most widely used, despite the fact that there are many more. It determines the usual discrepancy between predicted values and actual values [69]. It provides an estimate of how well the model can forecast the required quantity. It is given by Eq. (7).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x}_i)^2}{N}} \quad (7)$$

where:

N is the number of non-missing data points.

x_i is the actual observations time series.

$\hat{x}_i$ is the estimated time series.

The similarity results are given in Table 5.

Table 5. Similarity results

	Cosine Similarity	City Block Similarity
Item-based CF	1.50327465213	1.55417757528
User-based CF	1.49776385250	1.51726541940

We can deduce after observing the results that the best model is the one with the smallest value for RMSE. For our case it is user-item for the Cosine metric.

As a conclusion, we can state that memory-based models are simple to use and generate forecasts with acceptable accuracy. Since this sort of model evaluates the correlation between all users and things at each time, it is not scalable, i.e., it is not useful in a situation of a huge database. As a result, it does not address the issue of cold start [70], which arises when we begin with a new person or object about which we do not have sufficient knowledge.

To answer the scalability problem, we create Model Based models that we will deal with in the next section. And to address the problem of cold start, we use the Content based recommendation that we will also see later.

We calculated in Table 6 the performance with RMSE between the predicted matrix and the test matrix.

Table 6. Performance with RMSE between the predicted and the test matrices

		RMSE
SVD	k=50	1.50142530774
	k=75	1.48097179306
	k=80	1.47721123270
	k=100	1.46997911037
	k=110	1.46779703654
	k=125	1.47084637082
	k=150	1.47754794975
SGD	-	1.50060377979
ALS	Epoch 1/2	1.054704
	Epoch 2/2	0.799782

For the SVD technique, the prediction error measured with RMSE is 1.47, a smaller value than for the Memory based models, therefore a better performance. It should also be noted

that SVD takes considerably less time to run than Memory-based models.

Such a result of the RMSE, is influenced by the number of latent variables k, an important parameter of the SVD model. We therefore executed it by modifying each time the values of k to realize at the end that k = 100 is the optimal number of latent variables. So, for the performance measurement with RMSE (via the SGD technique), we calculate at each step the respective RMSE (see Figure 13). To obtain the error of the whole model, we calculate the average of the RMSEs at all stages (RMSE=1.50060377979).

The RMSE value obtained is higher than that obtained with SVD but remains better than the Memory based models. We did not limit ourselves to the final RMSE value, so we represented the error as a function of the number of steps previously set for the training and validation data. We notice that the two curves converge and the value of RMSE decreases with each new step.

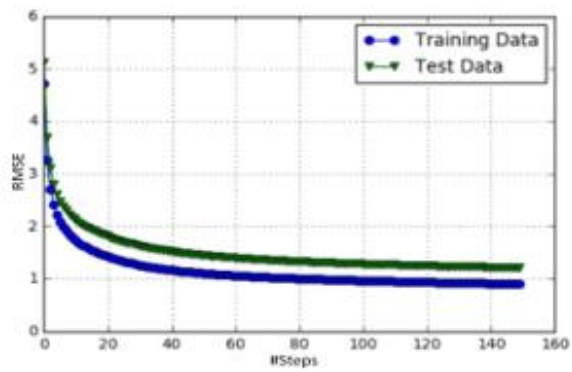

Figure 13. SGD learning curve

As the old model-based methods dealt with, we calculated the RMSE for the ALS model. As for SVD, ALS admits a definite number of steps through which the RMS decreases until it converges. In the case of our model, although the execution of ALS takes more time than SGD, we limited ourselves to 2 steps of the algorithm.

We notice that this error value for ALS is the best among all the other methods tested.

We can conclude that the best Collaborative Filtering model is ALS.

We created the User x Item recommendation matrix on our complete dataset after deciding that ALS was the best model.

To better look at the effectiveness of our model, we have implemented a set of functions that allow us to make the recommendation:

- **Recommendation of movies for a given user**: we seek to recommend to a user gives an example of movies that may interest him while associating each the predicted ratings.
- **Detect users similar to a given user**: We want to return users similar to a given use with the correlation between them.
- **Estimate the rating from a given user and movie**.
- **Calculation of correlations**: We have, in addition to the recommendations and the estimation of the ratings attributed to the movies, implemented functions that calculate the correlation between two users, two movies, and between a user and a movie.

4.2 Optimizing CF recommender systems

Let's look at the scenario where we want to compare movies based on the Genre variable (romantic, adventure, children, etc.). In our current dataset, the Gender variable is written in the form: first we must convert the Genre column into a matrix X containing 0 or 1: such that the rows of the matrix represent the movies (the movies) and the columns represent all the different types that we have in our dataset, and therefore X[i, j] = 1 if among the genres of the i^{th} movie, there is j^{th} Genre (see Tables 7 and 8).

We used the *vectorizer ()* function of the Python scikit-learn package to produce this matrix.

Table 7. Movie features

MovieID		Title	Genre	Year	Writer	Director	Cast	Rating	Runtimes	Countries	Language
0	**1**	Toy Story (1995)	Animation, Adventure, Comedy, Family, Fantasy	1995	John Lasseter, Pete Docter, Andrew Stanton, Joe R.	John Lasseter	Tom Hanks, Tim Alien, Don Rickles, Jim Vamey, Wal.	8.3	81	USA	English

Table 8. Genre matrix

Item	Comedy	Animation	Children	Family
1	1	0	1	0
2	0	1	0	0
3	1	1	1	0
4	0	0	0	1
5	1	0	0	1

Now the matrix we desired has been constructed, it is time to decide on a metric to determine how similar two objects are. There are many metrics to use when comparing two objects; in this instance, we choose cosine similarity. We ultimately get at our matrix in Table 9 after applying the cosine similarity metric, which calculates the similarity degree between all pairs of entries.

Several types of recommendations were proposed:

1) Recommendation based on the Gender variable: After obtaining the similarity matrix, for each movie we retrieve the indices of the movies that are the most similar to our selected one: the greater the cosine similarity metric, the greater the two movies are similar. Two movies are perfectly identical if cosine similarity=1. Let's test our function on the first movie: MovieID=1.

```
#data.query('MovieID==1|MovieID==2090|
MovieID==2355 | MovieID==1031 | MovieID==1030')
```

Table 9. Sample of similarities between all movie pairs

	0	1	2	3	4	5	6	7	8	9	…	889
0	1	0.600000	0.316228	0.258199	0.516398	0	0.316228	0.547723	0	0.259199		0.316228
1	0.600000	1	0	0	0.258199	0.447214	0	0.365148	0.516398	0.774597		0.316228
2	0.316228	0	1	0.816497	0.816497		0.500000	0.577350	0	0		0.500000
3	0.258199	0	0.816497	1	0.666667	0.288675	0.816497	0.707107	0	0		0.816497
4	0.516398	0.258199	0.816497	0.666667	1	0	0.408248	0.707107	0	0		0.408248
5	0	0.447214	0	0.288675	0	1	0.353553	0.204124	0.866025	0.577350		0.353553
6	0.316228		0.500000	0.816497	0.408248	0.353553	1	0.577350	0	0		1
7	0.547723	0.365148	0.577350	0.707107	0.707107	0.204124	0.577350	1	0	0.235702		0.577350
8	0	0.516398	0	0	0	0.866025	0	0	1	0.235702		0.577350
9	0.258199	0.774597	0	0	0	0.577350	0	0.235702	0.666667	1		0
10	0.258199	0	0.816497	1	0.666667	0.288675	0.816497	0.707107	0	0		0.816497
11	0.516398	0.258199	0.408248	0.333333	0.333333		0.408248	0.235702	0	0		0.408248
12	0.600000	0.400000	0	0.259199	0.258199	0.223607	0.316228	0.547723	0	0.258199		0.316228

The function returns movies classified according to the order of similarity. In our example, the function tells us that the movies closest to the *MovieID=1* in terms of Genre are: 2090, 2355, 1031, and 1030.

Let's check this in Table 10. We can see that the obtained result is very relevant, since the most similar movies to Toy Story movie is indeed The Rescuers (1977) movie as long as they are identical in terms of Genre.

2) Recommendation based on writer variable: We do the same thing as we did on the Gender variable.

```
#data.query('MovieID==1|MovieID==2355|
MovieID==1604' )[['MovieID','title','writer']] )
```

Let's check this in Table 11.

This result is very relevant since we notice that the movie Bug's Life which has the *MovieID=2355* has three writings in common with the Toy Story movie on which we did our research.

3) Recommendation based on the director variable: We apply the same function as before we get:

```
#(data.query('MovieID==1|MovieID==2355' ))[['MovieID',
'title','director']]
```

Let's check this in Table 12.

Bug's Life and Toy Story movies both have John Lasseter as director.

4) Recommendation based on the variable Production companies: We apply the same function as before we obtain:

```
#data.query('MovieID==1|MovieID==2355|
MovieID==1020'))[['MovieID','title','Production
Companies']]
```

Let's check this in Table 13.

Bug's Life and Toy Story movies have exactly the same Production Companies, which proves that our results are very relevant. According to all these results, we notice that the movie with ID=2355 appears on all the results, as a movie very similar to our selected movie.

5) General recommendation: The idea here is to combine all the recommendations made previously, to deduce a final recommendation that will be even more efficient. For this, for each movie for which we want to search for similar movies, we have applied a function that creates a DataFrame that contains all the MovieID with their scores (of similarity with respect to the selected movie) for each recommendation. Table 14 presents an example of the obtained result.

In our case we can conclude that MovieID=2355 is the most similar to MovieID=1, in terms of Genre, Writer, Director, Countries, and Production Companies, since it has a total score of 4.37. This confirms our remark that we made earlier.

4.3 Optimizing hybrid recommender systems

To illustrate our idea, we take as an example the case where we want to retrieve the indices of similar movies to MovieID=1, by combining CF and CB based on Gender. We get the results shown in Table 15. Based on this result, one could conclude that MovieID=4886 is very similar to MovieID=1.

Table 10. Closest movies to the MovieID=1 in terms of Genre

	MovieID	Title	Genre	Year	Writer
0	1	Toy Story (1995)	Animation, Adventure, Comedy, Family, Fantasy	1995	John Lasseter, Pete Docter
754	1030	Pete's Dragon (1977)	Animation, Adventure, Comedy, Family, Fantasy, Musical	1977	Malcolm
755	1031	Bedknobs and Broomsticks (1971)	Animation, Adventure, Comedy, Family, Fantasy, Musical	1971	Ralph Wright, Ted Berman
1488	2090	The Rescuers (1977)	Animation, Adventure, Comedy, Family, Fantasy	1977	Margery
1688	2355	A Bug's Life (1998)	Animation, Adventure, Comedy, Family, Fantasy	1998	John Lasseter, An- drew Stanton

Table 11. Closest movies to the MovieID=1 in terms of Writer

	MovieID	Title	Writer
0	1	Toy Story (1995)	John Lasseter, Pete Docter, Andrew Stanton, Joe Ranft, Joss Whedon, Andrew Stanton, Joel Cohen, Alec Sokolow
1150	1604	Money Talks (1997)	Joel Cohen, Alec Sokolow
1688	2355	A Bug's Life (1998)	John Lasseter, Andrew Stanton, Joe Ranft, Andrew Stanton, Don McEnery, Bob Shaw

Table 12. Closest movies to the MovieID=1 in terms of Director

	MovieID	Title	Director
0	1	Toy Story (1995)	John Lasseter
1688	2355	A Bug's Life (1998)	John Lasseter, Andrew Stanton

Table 13. Closest movies to the MovieID=1 in terms of Production Companies

	MovieID	Title	Production Companies
0	1	Toy Story (1995)	Pixar Animation Studios, Walt Disney Pictures
745	1020	Money Talks (1997)	Walt Disney Pictures
1688	2355	A Bug's Life (1998)	Pixar Animation Studios, Walt Disney Pictures

Table 14. General recommendation

Score_Genre	MovieID	Score_Writer	Score_Director	Score_Countries	Score_Production_Companies	Score_All	Score_Genre
1	1	2355	0.67082	0.707107	1.0	1	4.377927
24	0.8	1282	0	0	1.0	0.707107	2.507107
73	0.670820	1020	0	0	1.0	0.707107	2.377927
11	0.845154	1566	0	0	1.0	0.5	2.345154

Table 15. Indices of similar movies to MovieID=1 by combining CF and CB based solely on Gender

	Score_CB	MovieID	Score_CF	Score_All
5	1.0	4886	0.983677	1.983677
4	1.0	3114	0.982578	1.982578
0	1.0	2294	0.962589	1.962589
8	1.0	136016	0.938667	1.938667

The applied optimizations enable recommender systems to provide meaningful and personalized suggestions, even in scenarios with limited historical interaction data. The adaptability of these approaches ensures that recommendations become increasingly accurate and tailored as the system learns from user feedback and interactions over time. In addressing the cold start problem, CB approaches demonstrate effectiveness by leveraging item features and, in some cases, user-provided information. While ALS models have been successful in CF scenarios, they face challenges when dealing with sparse data and new items or users.

The choice between CB approaches and CF models like ALS depends on the specific characteristics of the recommendation scenario, including the availability of historical data and the nature of the items or users involved.

The proposed optimal hybrid approach combines the strengths of both methods and it provides a robust solution, ensuring effective recommendations in diverse situations.

5. SOLUTION'S ESTABLISHMENT

The different steps of the solution's establishment are as follows:

- Step 1 - Load and parse the data (RDD creation) (see Figure 14).
- Step 2 - CF: we used the Spark MLlib library which contains the implementation of the CF method using the ALS method.
- Step 3 - Focus recommendation on lower rated movies: one of the big problems with recommender systems is recommending always the 20% of the base that represents the most popular movies. Since they are the the more rated, they will be the most correlated with the other movies.
- Step 4 - Create a new user with ratings and merge this new RDD with the old database: we used this function to add users to the dataset, or to update the dataset notes if we have a new review event.
- Step 5: For a given user, we displayed the five movies most likely to please him (among the movies that have never rated). We used this function in the application to generate recommendations for users.
- Step 6: For a given user and a given movie, we estimated the rating and then compare with the real note. We used this function in the application to display a user on an establishment page, the predicted rating thanks to our model, in order to encourage him to give his own rating.

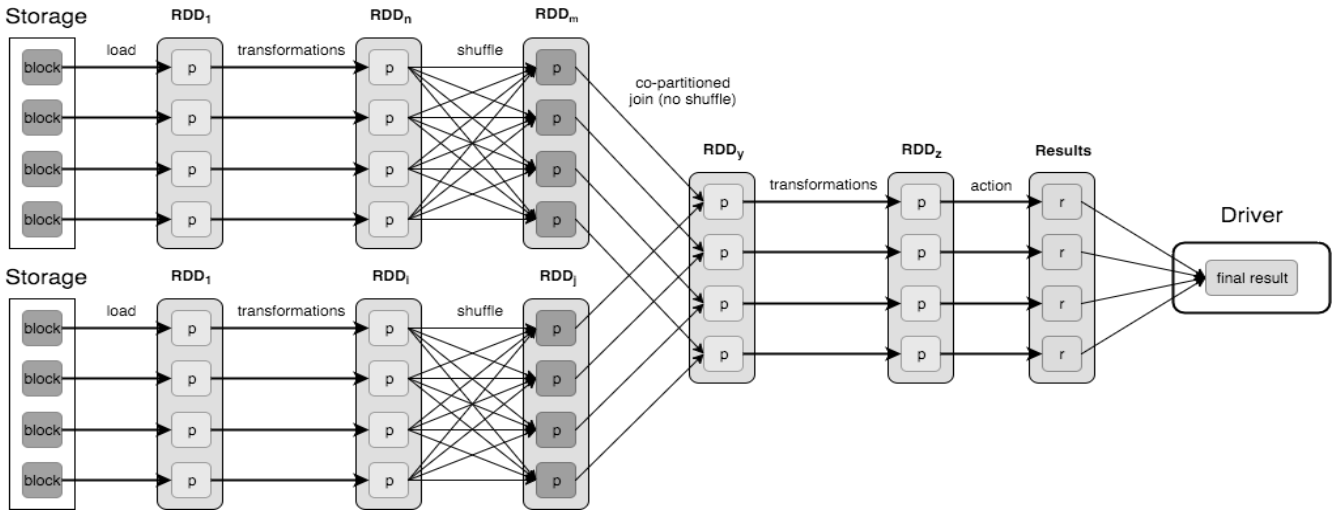

Figure 14. Resilient distributed dataset creation

6. DISCUSSION

Model-based CF approaches address data sparsity by leveraging mathematical models to infer latent factors that capture the underlying patterns in user-item interactions. These methods aim to overcome the challenges posed by sparse datasets, where users have interacted with only a small fraction of the available items. While model-based CF approaches offer advantages in handling data sparsity, they also have some limitations.

Matrix factorization techniques, such as SVD or ALS, decompose the user-item interaction matrix into latent factors. By representing users and items in a lower-dimensional latent space, these models can capture hidden patterns and relationships. These techniques are effective in handling data sparsity because it enables the model to fill in missing entries in the user-item interaction matrix. The inferred latent factors help predict how a user would interact with unrated items based on their preferences and the characteristics of those items.

Regularization also helps control the complexity of the model, reducing the risk of fitting noise in sparse datasets. It encourages the model to generalize well to unseen data, improving its ability to make accurate predictions for items with limited interactions.

In conclusion, model-based CF approaches effectively address data sparsity by leveraging mathematical models and regularization techniques. However, they are not immune to limitations, including challenges in handling the cold start problem and sensitivity to hyperparameters. Striking a balance between model complexity and interpretability is crucial in designing effective and scalable model-based CF systems for recommendation tasks.

7. CONCLUSIONS

Integrating different recommendation techniques seamlessly is not always straightforward. Combining CF and CB requires careful consideration of how these components interact, and the optimization process might be intricate.

For memory-based CF, the models generate good results, but they do not solve the problems of Cold Start -no information on users and movies-, sparsity, and scalability. For model-based CF, they solved the sparsity problem.

Indeed, hybrid recommender systems often rely on CF methods, which can face difficulties in handling sparse datasets. Limited user-item interactions make it challenging to accurately infer user preferences and item similarities. Moreover, the cold start problem persists in hybrid systems, particularly for new users or items. CB approach helps in recommending new items, but the lack of historical interaction data for CF can limit the system's ability to provide personalized suggestions.

The interpretability of hybrid models can also be limited. As the system combines CF and CB techniques, understanding the reasoning behind specific recommendations becomes challenging, potentially affecting user trust and satisfaction. Addressing these limitations and challenges requires a holistic approach, involving a careful balance between model complexity, scalability, adaptability, and user satisfaction. Regular monitoring, updating models based on user feedback, and experimenting with different algorithms and features are essential for optimizing hybrid movie recommender systems effectively. These two axes open on our future research work. This open on our future research work.

REFERENCES

[1] Forestiero, A. (2022). Heuristic recommendation technique in internet of things featuring swarm intelligence approach. Expert Systems with Applications, 187: 115904. https://doi.org/10.1016/j.eswa.2021.115904

[2] Rhanoui, M., Mikram, M., Yousfi, S., Kasmi, A., Zoubeidi, N. (2022). A hybrid recommender system for patron driven library acquisition and weeding. Journal of King Saud University - Computer and Information Sciences, 34(6, Part A): 2809-2819. https://doi.org/10.1016/j.jksuci.2020.10.017

[3] Tawfiq, F., Rahma, A.M.S., Abdulwahab, H.B. (2021). An e-commerce recommendation system based on dynamic analysis of customer behavior. Sustainability, 13(19). https://doi.org/10.3390/ su131910786

[4] Salampasis, M., Katsalis, A., Siomos, T., Delianidi, M., Tektonidis, D., Christantonis, K., Kaplanoglou, P., Karaveli, I., Bourlis, C., Diamantaras, K. (2023). A

flexible session-based recommender system for ecommerce. Applied Sciences, 13(5). https://doi.org/ 10.3390/app13053347

[5] Ben Schafer, J., Konstan, J., Riedl, J. (1999). Recommender systems in e-commerce. In Proceedings of the 1st ACM conference on Electronic commerce. Association for Computing Machinery, New York, NY, USA, pp. 158-166. https://doi.org/10.1145/336992.337035

[6] Schafer, J. B., Konstan, J., Riedl, J. (1999). Recommender systems in e-commerce. In 1st ACM Conference on Electronic Commerce, pp. 158-166. https://doi.org/10.1145/336992.337035

[7] Murugappan, V.M.R., Rodriguez, C., Navarro Depaz, C., Concha, U.R., Pandey, B., S. Kharat, R., Marappan, R. (2023). Machine learning based recommendation system for web-search learning. Telecom, 4: 118-134. https://doi.org/10.3390/telecom4010008

[8] Carrer-Neto, W., Hernández-Alcaraz, M.L., Valencia García, R., García-Sánchez, F. (2012). Social knowledge-based recommender system. application to the movies domain. Expert Systems with Applications, 39(12): 10990-11000. https://doi.org/10.1016/j.eswa.2012.03.025

[9] Burke, R. (2007). Hybrid web recommender systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), The Adaptive Web: Methods and Strategies of Web Personalization. Springer Berlin Heidelberg. pp. 377-408. https://doi.org/10.1007/978-3-540-72079- 9_12

[10] Schedl, M., Gómez, E., Lex, E. (2022). Retrieval and recommendation systems at the crossroads of artificial intelligence, ethics, and regulation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 3420-3424. https://doi.org/10.1145/3477495.3532683

[11] Mukund, N., Thakur, S., Abraham, S., Aniyan, A.K., Mitra, S., Philip, N.S., Vaghmare, K., Acharjya, D.P. (2018). An information retrieval and recommendation system for astronomical observatories. The Astrophysical Journal Supplement Series, 235(1): 22. https://doi.org/10.3847/1538-4365/aaadb2

[12] Mishra, R., & Rathi, S. (2022). Enhanced DSSM (deep semantic structure modelling) technique for job recommendation. Journal of King Saud University - Computer and Information Sciences, 34(9): 7790-7802. https://doi.org/10.1016/j.jksuci. 2021.07.018

[13] Ntioudis, D., Masa, P., Karakostas, A., Meditskos, G., Vrochidis, S., Kompatsiaris, I. (2022). Ontologybased personalized job recommendation framework for migrants and refugees. Big Data and Cognitive Computing, 6(4). https://doi.org/10.3390/bdcc6040120

[14] Junior, J., Vilasbôas, F., De Castro, L. (2020). The influence of feature selection on job clustering for an E-recruitment recommender system. https://doi.org/ 10.1007/978-3-030-61534-5_16

[15] Lee, Y., Lee, Y.C., Hong, J., Kim, S.W. (2017). Exploiting job transition patterns for effective job recommendation. In IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2414-2419. https://doi.org/10.1109/SMC.2017.8122984

[16] Reusens, M., Lemahieu, W., Baesens, B., Sels, L. (2017). A note on explicit versus implicit information for job recommendation. Decis. Support Syst., 98(100): 26-35. https://doi.org/10.1016/j.dss.2017.04.002

[17] Liu, K., Shi, X., Kumar, A., Zhu, L., Natarajan, P. (2016). Temporal learning and sequence modeling for a job recommender system. Recommender Systems Challenge. https://doi.org/10.1145/2987538. 2987540

[18] Alsaif, S.A., Sassi Hidri, M., Eleraky, H.A., Ferjani, I., Amami, R. (2022). Learning-based matched representation system for job recommendation. Computers, 11(11). https://doi.org/10.3390/computers11110161

[19] Alsaif, S.A., Hidri, M.S., Ferjani, I., Eleraky, H.A., Hidri, A. (2022). NLP-based bi-directional recommendation system: Towards recommending jobs to job seekers and resumes to recruiters. Big Data and Cognitive Computing, 6(4): 147. https://doi.org/10.3390/bdcc6040147

[20] Marchand, A., Marx, P. (2020). Automated product recommendations with preference-based explanations. Journal of Retailing, 96(3): 328-343. https://doi. org/https://doi.org/10.1016/j.jretai.2020.01.001

[21] Falconnet, A., Coursaris, C. K., Beringer, J., Van Osch, W., Sénécal, S., Léger, P.M. (2023). Improving user experience with recommender systems by informing the design of recommendation messages. Applied Sciences, 13(4). https://www.mdpi.com/2076-3417/13/4/2706/xml#.

[22] Kim, J., Choi, I., Li, Q. (2021). Customer satisfaction of recommender system: Examining accuracy and diversity in several types of recommendation approaches. Sustainability, 13(11). https://www.mdpi.com/2071-1050/13/11/6165.

[23] Guia, M., Silva, R., Bernardino, J. (2019). A hybrid ontology-based recommendation system in ecommerce. Algorithms, 12: 239. https://doi.org/10. 3390/a12110239

[24] Jayalakshmi, S., Ganesh, N., Cep, R., Senthil Murugan, ˇ J. (2022). Movie recommender systems: Concepts, methods, challenges, and future directions. Sensors, 22(13). https://doi.org/10.3390/s22134904

[25] Kavi Priya, S., Manonmani, T., Dharshana, N., Ragaanasuya, K. (2022). Movie recommendation system with hybrid collaborative and content-based filtering using convolutional neural network. International journal of health sciences, 6(S8): 5357-5372. https://sciencescholar.us/journal/index.php/ijhs/ article/view/13454.

[26] Salmani, S., Kulkarni, S. (2021). Hybrid movie recommendation system using machine learning. In International Conference on Communication, Information & Computing Technology, pp. 1-10. https://doi.org/10.1109/ICCICT50803.2021.9510058

[27] Kumar, M., Yadav, D., Singh, A., Kr, V. (2015). A movie recommender system: MOVREC. International Journal of Computer Applications, 124: 7-11. http://dx.doi.org/10.5120/ijca2015904111

[28] Colace, F., Conte, D., Santo, M.D., Lombardi, M., Santaniello, D., Valentino, C. (2022). A content-based recommendation approach based on singular value decomposition. Connection Science, 34(1): 2158-2176. https://doi.org/10.1080/09540091.2022.2106943

[29] Pérez-Almaguer, Y., Yera, R., Alzahrani, A.A., Martínez, L. (2021). Content-based group recommender systems: A general taxonomy and further improvements. Expert Systems with Applications, 184. https://doi.org/10.1016/j.eswa.2021.115444

[30] Joseph, A., Benjamin, J. (2022). Movie recommendation system using content-based filtering and cosine

similarity. National Conference on Emerging Computer Applications (NCECA), pp. 405-408. https://doi.org/10.5281/zenodo.6791117
[31] Javed, U., Shaukat, K., Hameed, I.A., Iqbal, F., Alam, T.M., Luo, S. (2021). A review of content-based and context-based recommendation systems. International Journal of Emerging Technologies in Learning (iJET), 16(3): 274-306. https://doi.org/10.3991/IJET.V16I03.18851
[32] Al Jobaer, A., Sanchi, B.B., Javed, A., Islam, M.S., Jameel, A.S.M.M. (2021). An advanced recommendation system by combining popularity-based and user-based collaborative filtering using machine learning. International Conference on Science & Contemporary Technologies (ICSCT), Dhaka, Bangladesh, pp. 1-5. https://doi.org/10.1109/ICSCT53883.2021.9642580
[33] Li, L., Wang, Z., Li, C., Chen, L., Wang, Y. (2022). Collaborative filtering recommendation using fusing criteria against shilling attacks. Connection Science, 34(1): 1678-1696. https://doi.org/10.1080/09540091.2022.2078280
[34] Liu, X., Li, S. (2021). Collaborative filtering recommendation algorithm based on similarity of co-rating sequence. In International Symposium on Electrical, Electronics and Information Engineering, pp. 458-463. https://doi.org/10.1145/3459104.3459180
[35] Wu, L. (2021). Collaborative filtering recommendation algorithm for MOOC resources based on deep learning. Complexity, 2021: 1-11. https://doi.org/10.1155/2021/5555226
[36] Hu, B., Long, Z. (2021). Collaborative filtering recommendation algorithm based on user explicit preference. In IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, pp. 1041-1043. https://doi.org/10.1109/ICAICA52286.2021.9498149
[37] Sun, N., Chen, T., Guo, W., Ran, L. (2021). Enhanced collaborative filtering for personalized e-government recommendation. Applied Sciences, 11: 12119. https://doi.org/10.3390/app112412119
[38] Çano, E. (2017). Hybrid recommender systems: A systematic literature review. Intelligent Data Analysis, 21: 1487-1524. https://doi.org/10.3233/IDA-163209
[39] Pande, C., Witschel, H.F., Martin, A. (2022). New hybrid techniques for business recommender systems. Applied Sciences, 12(10). https://doi.org/10.3390/app12104804
[40] Sacenti, J.A.P., Willrich, R., Fileto, R. (2018). Hybrid recommender system based on multi-hierarchical ontologies. In Proceedings of the 24th Brazilian Symposium on Multimedia and the Web, pp. 149-156. https://doi.org/10.1145/3243082.3243106
[41] Jannach, D., Pu, P., Ricci, F., Zanker, M. (2022). Recommender systems: Trends and frontiers. AI Magazine, 43(2): 145-150. https://doi.org/10.1002/aaai.12050
[42] Pajula, J., Kauppi, J.-P., Tohka, J. (2012). Intersubject correlation in FMRI: Method validation against stimulus-model based analysis. PloS one, 7: e41196. https://doi.org/10.1371/journal.pone.0041196
[43] Sammut, C., Webb, G.I. (2010). Tf–idf. In Encyclopedia of machine learning. Springer US. pp. 986-987. https://doi.org/10.1007/978-0-387- 30164-8_832
[44] Krishnapuram, B., Hartemink, A., Carin, L., Figueiredo, M.A.T. (2004). A Bayesian approach to joint feature selection and classifier design. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(9): 1105-1111. https://doi.org/10.1109/TPAMI.2004.55
[45] Quinlan, J.R. (1986). Induction of decision trees. Machine Learning, 1: 81-106. https://doi.org/10.1007/BF00116251
[46] Panteli, A., Boutsinas, B. (2023). Addressing the cold-start problem in recommender systems based on frequent patterns. Algorithms, 16(4). https://doi.org/10.3390/a16040182
[47] Jolliffe, I. (1986). Principal component analysis. Springer Verlag.
[48] Koren, Y., Bell, R., Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8): 30-37. https://doi.org/10.1109/MC.2009.263
[49] Landauer, T.K., Foltz, P.W., Laham, D. (1998). An introduction to latent semantic analysis. Discourse processes, 25(2-3): 259-284. https://doi.org/10.1080/01638539809545028
[50] Kabir, M.R., Zaiane, O. (2023). Classification by frequent association rules. In Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing, pp. 1217-1220. https://doi.org/10.1145/3555776.3577848
[51] Kyeong, S., Shin, J. (2022). Two-stage credit scoring using Bayesian approach. Journal of Big Data, 9. https://doi.org/10.1186/s40537-022-00665-5
[52] Sakib, N., Ahmad, R.B., Ahsan, M., Based, M.A., Haruna, K., Haider, J., Gurusamy, S. (2021). A hybrid personalized scientific paper recommendation approach integrating public contextual metadata. IEEE Access, 9: 83080-83091. https://doi.org/10.1109/ACCESS.2021.3086964
[53] Polatidis, N., Georgiadis, C. (2013). Mobile recommender systems: An overview of technologies and challenges. In 2013 2nd International Conference on Informatics and Applications, ICIA 2013, Lodz, Poland. https://doi.org/10.1109/ICoI
[54] Cai, X., Hu, Z., Zhao, P., Zhang, W., Chen, J. (2020). A hybrid recommendation system with many-objective evolutionary algorithm. Expert Systems with Applications, 159: 113648. https://doi.org/https://doi.org/10.1016/j.eswa.2020.113648
[55] Mgarbi, H., Chkouri, M.Y., Tahiri, A. (2023). Recommendation system: Technical study. In J. Kacprzyk, M. Ezziyyani, & V. E. Balas (Eds.), International Conference on Advanced Intelligent Systems for Sustainable Development, pp. 265-271. Springer Nature Switzerland.
[56] Zhou, M., Liu, Y. (2018/01). Hybrid recommendation and parallelization of movies based on spark. In Proceedings of the 2017 4th International Conference on Machinery, Materials and Computer (MACMC 2017), pp. 437-442. https://doi.org/10.2991/macmc17.2018.81
[57] Oh, J., Kim, S., Yun, S.-Y., Choi, S., Yi, M. (2019). A pipelined hybrid recommender system for ranking the items on the display. In RecSys Challenge '19: Proceedings of the Workshop on ACM Recommender Systems Challenge, pp. 1-5. https://doi.org/10.1145/3359555.3359565

[58] Bennett, J., Lanning, S. (2007). The Netflix prize. In Proceedings of the KDD Cup Workshop 2007, pp. 3-6. http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf.

[59] Kirch, W. (Ed.). (2008). Pearson's correlation coefficient. In Encyclopedia of Public Health, pp. 1090-1091. Springer Netherlands. https://doi.org/10.1007/978- 1-4020-5614-7_2569

[60] Zarei, M.R., Moosavi, M.R. (2019). A memory-based collaborative filtering recommender system using social ties. In 4th International Conference on Pattern Recognition and Image Analysis (IPRIA), pp. 263-267. https://doi.org/10.1109/PRIA.2019.8786023

[61] Al-bashiri, H., Abdulhak, M., Romli, A., Kahtan, H. (2018). An improved memory-based collaborative filtering method based on the TOPSIS technique. PLOS ONE, 13: e0204434. https://doi.org/10.1371/journal.pone.0204434

[62] Li, B., Han, L. (2013). Distance weighted cosine similarity measure for text classification. In H. Yin, K. Tang, Y. Gao, F. Klawonn, M. Lee, T. Weise, B. Li, & X. Yao (Eds.), Intelligent Data Engineering and Automated Learning – Ideal 2013, pp. 611-618. Springer Berlin Heidelberg.

[63] Wang, Z., Chen, J., Hu, J. (2022). Multi-view cosine similarity learning with application to face verification. Mathematics, 10(11). https://doi.org/10.3390/math10111800

[64] Mercioni, M.A., Holban, S. (2019). A survey of distance metrics in clustering data mining techniques. In Proceedings of the 3rd International Conference on Graphics and Signal Processing, pp. 44-47. https://doi.org/10.1145/3338472.3338490

[65] Pujahari, A., Sisodia, D.S. (2020). Model-based collaborative filtering for recommender systems: An empirical survey. In 2020 First International Conference on Power, Control and Computing Technologies (ICPC2T), Raipur, India, pp. 443-447. https://doi.org/10.1109/ICPC2T48082.2020.9071454

[66] Lathauwer, L.D., Moor, B.D., Vandewalle, J. (2000). A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applications, 21(4): 1253-1278. https://doi.org/10.1137/S0895479896305696

[67] Bottou, L. (2012). Stochastic Gradient Descent tricks. Berlin, Heidelberg:Springer Berlin Heidelberg, pp. 421–436,.https://doi.org/10.1007/978-3-642-35289-825

[68] Takács, G., Tikk, D. (2012). Alternating least squares for personalized ranking. In Proceedings of the Sixth ACM Conference on Recommender Systems, pp. 83-90. https://doi.org/10.1145/2365952.2365972

[69] Chai, T., Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? – Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 7(3): 1247-1250. https://doi.org/10.5194/gmd-7-1247-2014

[70] Herce-Zelaya, J., Porcel, C., Tejeda-Lorente, Á., BernabéMoreno, J., Herrera-Viedma, E. (2023). Introducing CSP dataset: A dataset optimized for the study of the cold start problem in recommender systems. Information, 14(1). https://doi.org/10.3390/info14010019