Design and Implementation of a Movie Recommendation System Based on Hybrid Recommendation Algorithms

1st Xiaowei Liu
College of Computer Science and Technology
Harbin Engineering University
Harbin, China
2020201513@harbeu.edu.cn

2nd Zhiwen Yu
College of Computer Science and Technology
Harbin Engineering University
Harbin, China
zhiwenyu@nwpu.edu.cn

***Abstract*—With the rapid expansion of online multimedia platforms, the ability to deliver personalized content becomes increasingly critical. Movie recommendation systems play a central role in helping users navigate large-scale content repositories and identify items aligned with their preferences. However, traditional single-method approaches such as collaborative filtering often struggle with data sparsity, cold-start issues, and limited robustness. To address these challenges, this paper presents a hybrid recommendation framework that integrates User-based Collaborative Filtering (UserCF) with Singular Value Decomposition (SVD). By combining neighborhood-level similarity signals with latent feature representations, the proposed model aims to improve the overall stability and applicability of recommendation outcomes. The hybrid model is trained on the MovieLens dataset and employs an adaptive weighting strategy to dynamically fuse the outputs of the two components. To demonstrate the framework's practical viability, we develop a fully functional movie recommendation system using Django, supported by MySQL for data management and Bootstrap for a responsive user interface. In addition, detailed movie metadata corresponding to MovieLens entries is scraped from the Douban platform to enrich the system's information content. Experimental observations and system deployment results indicate that the hybrid method operates reliably in real-world settings and provides a smooth user experience with consistent recommendation quality. Overall, the proposed framework bridges algorithmic design and practical deployment, offering a feasible solution for building personalized recommendation services in modern online media environments.**

***Index Terms*—movie recommendation, collaborative filtering, hybrid recommendation**

I. Introduction

The rapid proliferation of digital media platforms has fundamentally reshaped the way users interact with online information. As streaming services continue to grow, users are faced with the challenge of navigating massive collections of movies and related multimedia content. This phenomenon, commonly referred to as information overload, greatly reduces users' ability to efficiently identify content that aligns with their preferences. Recommender systems have therefore become indispensable tools for improving user experience by offering personalized content filtering and decision support [1].

Among various application domains, movie recommendation systems serve as a representative benchmark for evaluating recommendation algorithms due to the diversity of user behaviors and the richness of item metadata. Traditional recommendation methods are generally categorized into content-based filtering and collaborative filtering (CF). Content-based methods rely on movie attributes such as genre, cast, or keywords, making them effective when rich metadata is available. However, they often fail to capture hidden user preferences or complex behavioral patterns [2]. In contrast, CF methods infer user interests by examining similarities among users or among items, achieving strong performance in many practical scenarios. Despite their popularity, CF approaches suffer from well-known limitations, including data sparsity, cold-start problems, and reduced performance when rating matrices are highly incomplete [3].

To address these deficiencies, model-based methods, particularly matrix factorization techniques such as Singular Value Decomposition (SVD), have been introduced to uncover latent factors governing user–item interactions. These techniques can mitigate sparsity by capturing implicit patterns that cannot be detected through simple similarity computations [4]. Nevertheless, SVD alone is limited by its dependence on global factorization and its reduced interpretability compared to similarity-based CF [5]. This creates an important research gap: while CF excels at identifying local preference structures, SVD provides a more global and generalized understanding of user behavior. A natural and effective direction is therefore to integrate these two complementary perspectives through a well-designed hybrid recommendation framework.

Motivated by this observation, this paper proposes a hybrid movie recommendation algorithm that combines user-based CF with SVD through an adaptive weighting strategy [6]. The hybrid approach leverages the interpretability and neighborhood precision of CF while benefiting from the robustness and dimensionality reduction offered by SVD. By integrating both models, the system achieves superior performance in recommending movies to users with both dense and sparse rating histories [7]. Furthermore, to demonstrate the practical

applicability of the proposed algorithm, this work develops a fully functional web-based movie recommendation system using Django, MySQL, and Bootstrap. The system incorporates essential components including data ingestion, rating storage, personalized recommendation generation, user interface rendering, and session management [8].

A notable feature of this work is the construction of a comprehensive data enrichment pipeline. Although MovieLens provides reliable user–movie rating data, it lacks detailed movie descriptions. To address this issue, we implemented a large-scale IMDb web crawler to extract metadata including poster links, cast, directors, summaries, and release dates. The extracted content is subsequently localized using the Baidu Translate API, enabling the system to provide bilingual movie descriptions [9]. This enriched dataset facilitates not only improved personalization but also enhances user engagement by presenting semantically meaningful movie information.

To evaluate the effectiveness of the hybrid model, extensive experiments were conducted on the MovieLens latest-small dataset. Results show that the proposed hybrid approach significantly reduces RMSE and yields higher accuracy in top-N recommendations compared with standalone CF and SVD models [10]. In addition, the deployed system demonstrates stable performance under concurrent usage, providing real-time recommendations suitable for practical web environments [11].

The key contributions of this paper are as follows.

- We propose a hybrid recommendation framework that integrates user-based Collaborative Filtering (UserCF) with SVD.The framework balances local similarity modeling from CF with global latent factor extraction from SVD. This hybrid strategy enhances overall recommendation accuracy compared with traditional single-method approaches.
- We develop a complete end-to-end movie recommendation system built with Django, MySQL, and Bootstrap.The system includes modules for user authentication, movie browsing, personalized recommendation generation, and data management. Its modular architecture ensures smooth interaction between backend algorithms and frontend presentation.
- We validate the hybrid recommendation framework through deployment and testing within the implemented system. Practical application results show that the hybrid approach delivers consistently accurate recommendations and provides a smooth real-time user experience.

II. Related Work

Research on recommender systems has developed extensively over the past several decades, evolving from early heuristic techniques to sophisticated algorithmic frameworks. Foundational work by Resnick and Varian first formalized the concept and significance of recommender systems in mitigating information overload in digital environments. Subsequent surveys, such as the influential review by Adomavicius and Tuzhilin, provided a comprehensive taxonomy of major recommendation paradigms and highlighted persistent challenges such as data sparsity, scalability, and evaluation methodology. These studies laid the groundwork for the rapid expansion of personalized recommendation technologies across various domains [12].

A. Collaborative Filtering

Collaborative Filtering (CF) remains one of the most widely adopted approaches in recommender systems. User-based and item-based CF methods determine preference patterns based on rating correlations among users or items. The item-based CF model introduced by Sarwar et al. demonstrated improved scalability by leveraging item co-occurrence structures. Schafer et al. further examined CF applications within e-commerce scenarios, where recommendation quality directly influences user engagement and platform profitability [13], [14], [20]. In addition, earlier domestic studies also explored CF from an implementation perspective, demonstrating its practical usability in medium-scale systems. Despite these strengths, CF suffers from limitations such as data sparsity and cold-start conditions, motivating research on enhanced variants and hybrid strategies [15], [16], [21].

B. Matrix Factorization and SVD Methods

Matrix factorization, especially Singular Value Decomposition (SVD), has emerged as a dominant technique for modeling user–item interactions by uncovering latent semantic structures. The landmark work by Koren et al. illustrated the effectiveness of SVD-based factorization in the Netflix Prize competition, showing substantial improvements over similarity-based models. Additional academic research has extended SVD to more robust formulations capable of handling incomplete and high-dimensional rating matrices [17], [22]. These methods generally achieve higher predictive accuracy than classical CF due to their ability to capture latent relationships and reduce noise. However, they may lack interpretability and can require careful parameter tuning, leading many studies to explore hybrid combinations with neighborhood-based approaches [23].

C. Hybrid Recommendation Models

Hybrid recommender systems combine multiple algorithms to leverage their respective advantages and mitigate individual weaknesses. Systematic reviews highlight that hybridization strategies—including weighted fusion, switching mechanisms, and feature augmentation—typically yield more stable and accurate recommendation performance across diverse datasets [18]. Empirical findings show that combining neighborhood models with matrix factorization allows the system to utilize both local similarity information and global latent factor structures. These findings motivate the hybrid CF–SVD strategy adopted in this work, which aims to improve robustness under sparse rating distributions while maintaining high interpretability [24].

D. Datasets and System Implementations

Benchmark datasets such as MovieLens, maintained by GroupLens Research, serve as widely accepted testing grounds for recommendation algorithms due to their clean data structure and suitability for both CF-based and factorization-based approaches [25], [26]. Practical recommendation platforms also rely heavily on robust web frameworks and databases. Modern implementations commonly employ technologies such as Django for backend logic and MySQL for persistent data management, supported by the extensive documentation provided by both communities. These foundational tools enable researchers to validate algorithms in real-world system environments, bridging the gap between theoretical models and deployable applications [19].

III. Algorithm Design

This section presents the hybrid recommendation algorithm adopted in the system. To provide accurate and robust movie recommendations, the framework integrates concepts from traditional CF and model-based matrix factorization. Specifically, we introduce user-based CF, item-based CF, and SVD-based matrix factorization. Although both user-based and item-based CF were implemented and evaluated during development, experimental results later demonstrated that user-based CF consistently outperformed item-based CF in the target scenario [27], [28]. Therefore, the final hybrid model is composed of user-based CF and SVD, while item-based CF is not used in the final fusion stage. The detailed comparison will be discussed in the Experiments section.

A. User-based Collaborative Filtering

User-based Collaborative Filtering predicts a user's preference based on the rating behaviors of similar users [29]. The assumption is that users with similar historical preferences tend to show similar future interests, as shown in Figure 1. To quantify the similarity between users, the Pearson correlation coefficient is used, as it captures rating tendencies while removing individual scale biases.

The similarity between users u and v is computed as:

$$\text{sim}(u,v) = \frac{\sum_{j\in I_{uv}}(r_{u,j}-\bar{r}_u)(r_{v,j}-\bar{r}_v)}{\sqrt{\sum_{j\in I_{uv}}(r_{u,j}-\bar{r}_u)^2}\sqrt{\sum_{j\in I_{uv}}(r_{v,j}-\bar{r}_v)^2}}. \quad (1)$$

where I_{uv} is the set of items jointly rated by users u and v, $r_{u,j}$ and $r_{v,j}$ denote the ratings of u and v on item j, $\bar{r}_u$ and $\bar{r}_v$ are their average ratings, and $\text{sim}(u,v)$ represents the Pearson similarity.

Once similarities are obtained, the top-k nearest neighbors $N_k(u)$ are selected, and the prediction for an unrated item i is computed as:

$$\hat{r}_{u,i}^{CF} = \bar{r}_u + \frac{\sum_{v\in N_k(u)}\text{sim}(u,v)\,(r_{v,i}-\bar{r}_v)}{\sum_{v\in N_k(u)}|\text{sim}(u,v)|}. \quad (2)$$

where $N_k(u)$ is the set of top-k users most similar to user u, $r_{v,i}$ is the rating of neighbor v on item i, $\bar{r}_u$ is the baseline rating of user u, and $\hat{r}_{u,i}^{CF}$ denotes the predicted score.

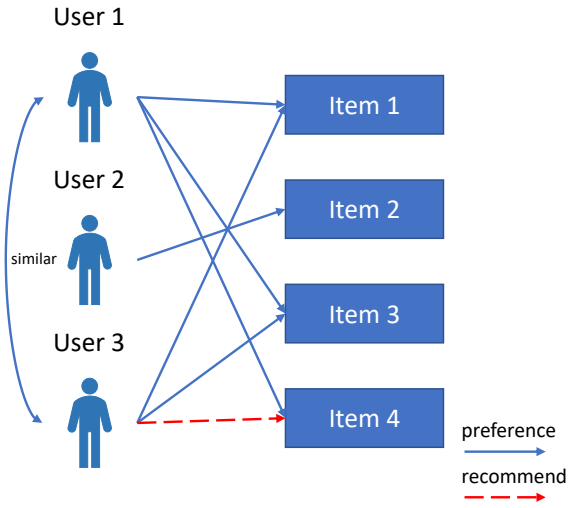

Fig. 1. Schematic diagram of user based CF algorithm.

User-based CF captures personalized and local behavioral similarity, making it particularly effective when users have rich historical ratings.

B. Item-based Collaborative Filtering

In contrast to user-based CF, item-based CF focuses on the relationships between items, as shown in Figure 2. The core idea is that items frequently co-rated by users may share similar characteristics. When predicting a rating for user u and item i, the algorithm identifies items similar to i that user u has rated and uses these to derive the prediction [30], [31].

Item similarity between items i and j is computed as:

$$\text{sim}(i,j) = \frac{\sum_{u\in U_{ij}}(r_{u,i}-\bar{r}_i)(r_{u,j}-\bar{r}_j)}{\sqrt{\sum_{u\in U_{ij}}(r_{u,i}-\bar{r}_i)^2}\sqrt{\sum_{u\in U_{ij}}(r_{u,j}-\bar{r}_j)^2}}. \quad (3)$$

where U_{ij} is the set of users who rated both items i and j, $r_{u,i}$ and $r_{u,j}$ denote user u's ratings for the two items, $\bar{r}_i$ and $\bar{r}_j$ are their average ratings, and $\text{sim}(i,j)$ represents the Pearson similarity between items.

Given the computed similarities, the prediction for user u on item i is:

$$\hat{r}_{u,i}^{ItemCF} = \bar{r}_i + \frac{\sum_{j\in S_k(i)}\text{sim}(i,j)\,(r_{u,j}-\bar{r}_j)}{\sum_{j\in S_k(i)}|\text{sim}(i,j)|}. \quad (4)$$

where $S_k(i)$ is the set of top-k items most similar to item i, $r_{u,j}$ is the rating of user u on item j, $\bar{r}_i$ is the baseline score of item i, and $\hat{r}_{u,i}^{ItemCF}$ denotes the predicted rating.

Item-based CF is often more stable than user-based CF when user behaviors are sparse. However, in our experimental environment, user-based CF consistently generated lower prediction error and higher recommendation precision. Therefore, ItemCF was implemented but not included in the final hybrid model.

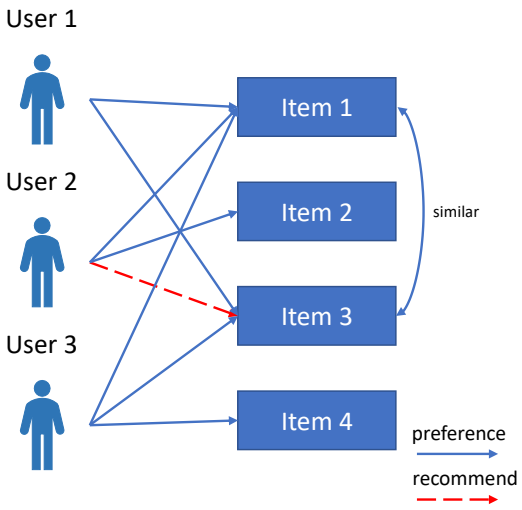

Fig. 2. Schematic diagram of item-based CF algorithm.

C. SVD-based Matrix Factorization

Matrix factorization maps users and items into a shared latent factor space. Unlike Collaborative Filtering methods that rely on co-rating overlaps, SVD captures global patterns embedded in the rating matrix, enabling it to handle sparse data and users with limited rating history effectively.

The rating matrix R is approximated as:

$$R \approx U\Sigma V^T. \tag{5}$$

where U and V denote the user and item latent factor matrices, respectively, and Σ contains the singular values representing the importance of latent dimensions.

The predicted rating for user u on item i is computed as:

$$\hat{r}_{u,i}^{SVD} = \mu + b_u + b_i + p_u^T q_i. \tag{6}$$

where μ is the global average rating, b_u and b_i are user and item biases, and p_u, q_i are the latent factor vectors for user u and item i, respectively.

Through the combination of global bias modeling and latent factor interactions, SVD learns smooth and generalizable patterns such as genre affinity and style preferences that cannot be captured by simple neighborhood-based methods.

D. Score Normalization and Weighted Fusion

Because the prediction scales of CF and SVD differ, min–max normalization is applied to align their outputs before fusion:

$$\tilde{r}_{u,i} = \frac{\hat{r}_{u,i} - r_{\min}}{r_{\max} - r_{\min}}. \tag{7}$$

where $r_{\min}$ and $r_{\max}$ denote the minimum and maximum predicted scores of a given model, and $\tilde{r}_{u,i}$ is the normalized rating.

The final hybrid prediction integrates CF and SVD using a weighted scheme:

$$\hat{r}_{u,i}^{hybrid} = \alpha \tilde{r}_{u,i}^{CF} + (1-\alpha)\tilde{r}_{u,i}^{SVD}. \tag{8}$$

Algorithm 1 Hybrid CF–SVD Recommendation Algorithm

Require: Rating matrix R, target user u, candidate item set C, CF parameter k, SVD parameters $(f, epochs, lr, reg)$, fusion weight α
Ensure: Top-N recommended items for user u

```
Train SVD model offline to learn p_u, q_i
Precompute or periodically update user–user similarities
Retrieve top-k similar users N_k(u)
for each item i ∈ C not rated by u do
    Compute CF prediction r̂^CF_{u,i}
    Compute SVD prediction r̂^SVD_{u,i}
end for
Normalize r̂^CF and r̂^SVD to [0, 1]
for each item i ∈ C do
    Compute fused score:
        r̂^hybrid_{u,i} = α r̃^CF_{u,i} + (1 − α) r̃^SVD_{u,i}
end for
Rank candidate items by r̂^hybrid_{u,i}
return Top-N items
```

where $\alpha \in [0,1]$ determines the contribution of user-based CF, $\tilde{r}_{u,i}^{CF}$ and $\tilde{r}_{u,i}^{SVD}$ are the normalized predictions from the two models, and $\hat{r}_{u,i}^{hybrid}$ is the final fused rating.

During algorithm selection, experimental comparisons demonstrated that user-based CF consistently outperformed item-based CF in RMSE and Top-N recommendation metrics. Therefore, the hybrid model incorporates only user-based CF alongside SVD, while item-based CF is excluded [18], [19]. Detailed evaluations and performance justifications are presented in the Experiments section.

E. Hybrid CF–SVD Recommendation Algorithm

The complete workflow of the hybrid recommendation algorithm is summarized in Algorithm 1. The algorithm trains the SVD model offline, computes user similarities periodically, and generates predictions by combining two complementary models at query time.

F. Complexity Analysis

CF requires $O(|U||I_u|)$ similarity computation unless precomputed, while SVD training requires $O(|R|f \cdot epochs)$. The fusion step incurs negligible cost. Once trained, the hybrid model is efficient enough for real-time recommendation scenarios, since both CF and SVD predictions can be generated with linear complexity relative to the candidate set size.

IV. Experiments

A. Dataset

The MovieLens dataset used in this article is a widely used movie rating dataset created and maintained by the GroupLens research group, which is used to evaluate the recommendation performance of recommendation algorithms. This dataset contains user ratings and comments on movies. The commonly used versions in this dataset are MovieLens

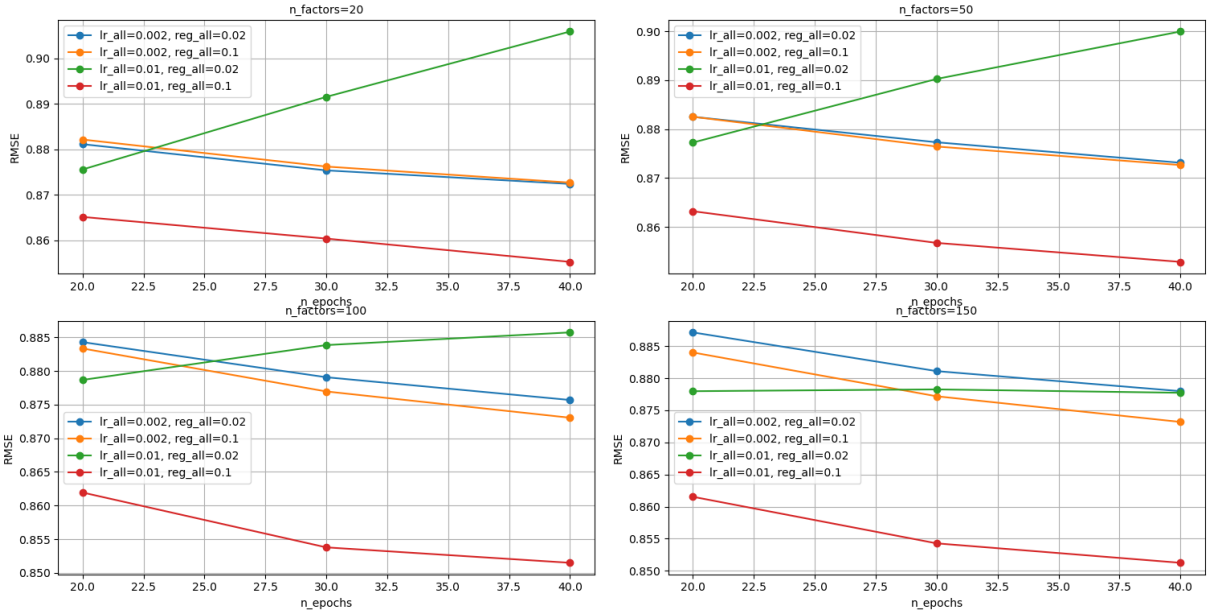

Fig. 3. Experimental results of SVD recommendation algorithm.

100k, MovieLens 1M, MovieLens 10M, MovieLens 20M, and MovieLens 25M, where the numbers represent the number of ratings in the corresponding dataset. This project uses the ml latest small dataset in MovieLens, which contains 610 users' ratings of 9742 movies, with a total of 100837 rating records. The dataset structure is shown in Table I.

TABLE I
STRUCTURE OF THE ML-LATEST-SMALL DATASET IN MOVIELENS

Data File	Size	Field	Description
movies	9742	movieId	Movie identifier
		title	Movie title
		genres	Movie genres
tags	3683	userId	User identifier
		movieId	Movie identifier
		tag	User-generated tag
		timestamp	Tagging timestamp
links	9742	movieId	Movie identifier
		imdbId	IMDB identifier
		tmdbId	TMDB identifier
ratings	100837	userId	User identifier
		movieId	Movie identifier
		rating	Rating score
		timestamp	Rating timestamp

The MovieLens dataset is widely used for evaluating recommendation systems, researching collaborative filtering algorithms, and related machine learning tasks. This dataset provides a standard benchmark for rigorous comparison and evaluation of different algorithms. Due to the rich user rating information provided by the MovieLens dataset, it is highly suitable for research on movie recommendation systems. We can construct a "user rating" matrix based on the user rating information in the dataset. By analyzing the user's historical rating records, we can build a relationship model between the user and the item, and predict the user's preference for unrated movies. By utilizing the MovieLens dataset, we can compare the accuracy, coverage, diversity, and other metrics of different algorithms to find the most suitable recommendation strategy for the application scenario.

B. Experimental Design

The implementation logic of the proposed recommendation algorithm is as follows. All user–movie rating data are first retrieved from the database to construct a user–item rating matrix. This rating matrix is then decomposed using SVD to obtain the user feature matrix, the singular value matrix, and the movie feature matrix. Based on these matrices, predicted ratings for every user–movie pair are computed through matrix multiplication. For the current user, all unrated movies are filtered and ranked according to the predicted scores, and the system finally returns the top-N movies together with their predicted ratings.

C. Result Analysis

When conducting experiments with the user-based collaborative filtering algorithm, different values of K were selected to determine which neighborhood size yields the best predictive accuracy. The recommendation performance was evaluated using the Root Mean Square Error (RMSE), where a lower RMSE indicates better predictive quality. The experimental results are shown in Figure 4.

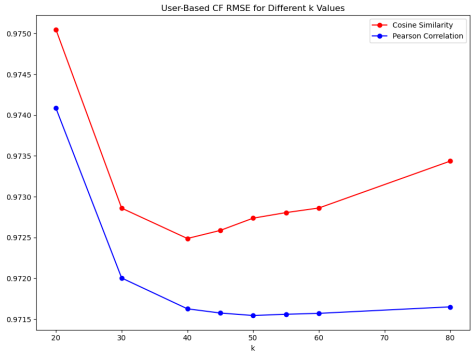

Fig. 4. Experimental results of user based CF recommendation algorithm.

According to the results, the optimal performance is achieved when $K = 50$, with similarity computed using the Pearson correlation coefficient.

For the SVD-based recommendation algorithm, experiments were conducted by varying the four main parameters: `n_factors` (the number of latent factors in matrix decomposition), `n_epochs` (the number of training iterations), `lr_all` (the learning rate), and `reg_all` (the regularization coefficient). RMSE was again used to evaluate performance. The experimental results show that the best performance is obtained when `n_factors` is set to 150, `n_epochs` to 40, `lr_all` to 0.01, and `reg_all` to 0.1. The corresponding results are illustrated in Figure 3.

Overall, the SVD-based recommendation algorithm demonstrates superior performance compared with the user-based collaborative filtering approach.

V. System Design and Implementation

A. Requirement Analysis

The core of this system is to use a hybrid recommendation algorithm to recommend movies that users may be interested in. When designing a movie recommendation system based on a hybrid recommendation algorithm, it is necessary to first clarify the main objectives and functions of the system. The system aims to provide users with personalized movie recommendations to enhance their viewing experience and improve user satisfaction. The system should have the following requirements:

1) New user creation. The system needs to implement new user creation. This includes the design of the user registration interface, where users need to fill in basic information, and the user creation interface should automatically detect whether the information filled in by the user is correct and handle exceptions correctly. After the user is created, their information will be stored in the system database.
2) User login. The system needs to implement user login functionality. Users can log in to the system using their previously registered username and password, so that the system can recognize the user's identity and provide personalized services. In the user login interface, an automatic check function should be implemented for the information filled in by the user, and abnormal behavior of filling in incorrect information should be reported.
3) Recommended popular movies. The system needs to implement the function of recommending popular movies. Popular movies can be determined by analyzing data such as the number of viewers and ratings, and displayed to users on the system homepage. Users can browse the currently hottest movies and click to enter the movie details interface to browse relevant information.
4) Movie list display. The system needs to implement the movie list display function. After the user logs in, the system should display a rich list of movies to the user for easy browsing and selection.
5) View movie details. The movie details browsing function requires a detailed movie information page, including the plot summary, cast list, director information, release date, etc. Users can obtain comprehensive information about movies on this page to better understand their content.
6) Movie recommendation. The movie recommendation function is one of the core functions of the entire system. The system needs to use recommendation algorithms to recommend movies that meet the preferences and behavior data of users. The recommendation results can be displayed on the system's homepage, personalized recommendation page, or specialized recommendation list.
7) Movie search. The movie search function needs to provide a convenient and fast search interface, where users can search for movies using keywords. The system should match the movie database based on the user's search criteria and return corresponding search results, so that the user can find the movie they are interested in.

B. System Design

Based on the system requirements analysis in the previous section of this article, we can create a four layer system, including the front-end interface layer, application logic layer, back-end service layer, and data storage layer.

The front-end interface layer is the interface for users to interact with the system, including various forms such as web pages and mobile devices. The interface layer is responsible for displaying the user interface and providing interactive functions to users, such as registration and login interface, movie recommendation page, movie list display page, movie details page, search page, etc. The front-end interface communicates with the application logic layer through HTTP requests, receives user operation requests, and passes them to the application logic layer for processing.

The application logic layer is the core part of the system, responsible for controlling the business logic and processes of the system. The application logic layer processes user requests, calls backend services for data processing, and returns the processing results to the frontend interface layer. The

application logic layer includes user registration and login logic, popular movie recommendation logic, movie list display logic, movie detail display logic, movie recommendation logic, movie search logic, etc.

The backend service layer provides the core functional services of the system, including user management services, movie information services, recommendation algorithm services, search services, etc. The user management service is responsible for storing, verifying, and managing user information, including functions such as user registration and login. The movie information service is responsible for obtaining and managing movie information, including functions such as querying and updating movie details. The recommendation algorithm service is the core part of the system, responsible for providing personalized movie recommendations to users based on their personal preferences and behavioral data using hybrid recommendation algorithms. The search service is responsible for matching the movie information database based on the user's search criteria and returning the corresponding search results.

The data storage layer is responsible for storing user data, movie data, and recommendation result data required by the system. Multiple databases including user information database, movie information database, recommendation result database, etc. are stored using relational databases. The user information database stores basic user information such as username, password, personal preferences, etc. The movie information database stores detailed information about movies, such as movie titles, directors, actors, plot summaries, ratings, etc. The recommendation result database stores the recommendation results generated by the recommendation algorithm, in order to quickly provide personalized recommendation services.

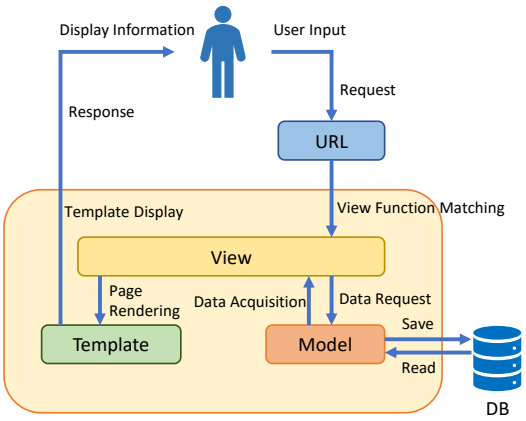

Fig. 5. Experimental results of user based CF recommendation algorithm.

C. System Implementation

The system was implemented on a Windows 11 environment using Python, MySQL, Bootstrap, and supporting tools such as PyCharm and Navicat. The system development uses the Django framework that adopts the MTV design pattern. The MTV mode clearly separates application logic, data model, and view logic, making the code easy to understand, maintain, and extend. Its structure is shown in Figure 5. The Model layer is responsible for defining and managing data structures. Template is the template layer responsible for generating and displaying user interfaces. View is the view layer responsible for processing user requests and generating responses. It serves as a bridge between the model layer and the template layer. The design of views effectively separates business logic and data logic, while also improving code reusability and testability.

The implemented system realizes all core functional modules defined during the design phase. The registration module validates input completeness, checks for existing accounts, applies secure password hashing, and stores qualified entries in the database. The login module authenticates user credentials and establishes stable session states. For content browsing, the movie list module adopts server-side pagination to efficiently handle large datasets, while the popular-movie module derives frequently rated or high-score items and displays them to enhance user engagement.

The movie-detail module provides complete metadata by querying the database through unique movie identifiers. The search module incorporates fuzzy keyword matching to retrieve movies relevant to user input. The recommendation module integrates collaborative filtering and SVD-based representations to produce hybrid recommendation lists tailored to individual preferences. Together, these modules form a coherent and user-oriented movie recommendation system that supports browsing, searching, and personalized content delivery. In addition, the system maintains efficient data access through optimized indexing strategies, ensuring low-latency responses even under high user load. These combined mechanisms create a seamless interaction experience and enhance the overall effectiveness of the recommendation pipeline.

VI. Conclusion

This work presents the development of a movie recommendation system built upon a hybrid recommendation framework that integrates user-based collaborative filtering with SVD-based matrix factorization. By preprocessing the MovieLens dataset, crawling and refining detailed metadata for 9,742 films from IMDB, a complete and structured movie information database was successfully constructed. On this foundation, the hybrid recommendation algorithm was implemented and demonstrated stable and effective recommendation performance. Furthermore, a functional web-based system was developed using Django and Bootstrap, enabling user registration, authentication, movie browsing, personalized recommendation, and keyword search. Although the system still exhibits limitations in interface design, interaction optimization, and recommendation response time, the implemented platform verifies the feasibility and practical value of the proposed hybrid recommendation approach and provides a basis for future enhancement.

References

[1] L. Xu, Z. Guan, and Y. Wu, "Enhancing movie recommendation systems with hybrid collaborative filtering, content-based filtering and SVD," In Proceedings of the 2nd International Conference on Machine Learning and Automation, CONF-MLA 2024, November 21, 2024, Adana, Turkey (p. 351). European Alliance for Innovation.

[2] S. Sharma and H. K. Shakya, "Hybrid recommendation system for movies using artificial neural network," Expert Systems with Applications, vol. 258, p. 125194, Dec. 2024.

[3] J. Bobadilla, J. Dueñas-Lerín, F. Ortega, and A. Gutiérrez, "Comprehensive evaluation of matrix factorization models for collaborative filtering recommender systems," Int. J. Interactive Multimedia & Artificial Intelligence, vol. 8, no. 6, pp. 15–23, Jun. 2024.

[4] Y. Yan, C. Moreau, Z. Wang, W. Fan, and C. Fu, "Transforming movie recommendations with advanced machine learning: a study of NMF, SVD, and K-means clustering," unpublished, 2024.

[5] Z. Wang, "A content-based collaborative filtering algorithm for movies and TV recommendation," Applied and Computational Engineering, vol. 15, pp. 83–91, Oct. 2023.

[6] Y. Guo, "Enhancing movie recommendation systems through CNN-based feature extraction and optimized collaborative filtering," in Applied & Computational Engineering, vol. 81, Nov. 2024.

[7] Y. Long, "Application of collaborative filtering in movie recommendation systems and improvements by hyperparameter tuning," in Applied & Computational Engineering, vol. 73, Jul. 2024, pp. 1–7.

[8] S. Sharma, G. Dubey, and H. K. Shakya, "Hybrid filtering techniques and improved SOM for next-generation film recommendations," Int. J. Intelligent Systems & Applications in Engineering, vol. 12, no. 3, pp. 3477–3484, Mar. 2024.

[9] A. Yadav, G. Srivastava, and S. Kumar, "A hybrid approach to movie recommendation system," J. Manage. Serv. Sci., vol. 4, no. 1, pp. 1–14, Apr. 2024.

[10] Y. Tang, "Movie recommendation system based on collaborative filtering network," in Proc. 2nd Int. Conf. Machine Learning & Automation (CONF-MLA), Nov. 2024, pp. 113–119.

[11] Y. Wang, "Movie recommendation system based on SVD collaborative filtering," in Proc. 2nd Int. Conf. Artificial Intelligence, Big Data & Algorithms (CAIBDA), Nanjing, China, Jun. 2022.

[12] Shaina Raza, Mizanur Rahman, Safiullah Kamawal, Armin Toroghi, Ananya Raval, Farshad Navah, Amirmohammad Kazemeini, "A comprehensive review of recommender systems: Transitioning from theory to practice," unpublished (arXiv), 2024.

[13] Xubin Ren, Wei Wei, Lianghao Xia, Chao Huang, "A comprehensive survey on self-supervised learning for recommendation," unpublished (arXiv), 2024.

[14] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Jundong Li, Zi-Lin Huang, "Self-supervised learning for recommender systems: A survey," IEEE Transactions on Knowledge and Data Engineering , vol.36, no.1, pp.335–355, Jan.2024.

[15] S. Mhammedi, H. El Massari, N. Gherabi, A. Mohamed, "CF recommender system based on ontology and nonnegative matrix factorization," unpublished (arXiv), 2024.

[16] O. A. S. Ibrahim, et al. , "Revisiting recommender systems: An investigative survey," Neural Computing and Applications , in press (2025).

[17] Jesús Bobadilla, Jorge Dueñas-Lerín, Fernando Ortega, Abraham Gutiérrez, "Comprehensive evaluation of matrix factorization models for collaborative filtering recommender systems," International Journal of Interactive Multimedia & Artificial Intelligence , vol. 8, no. 6, pp. 15–23, June 2024.

[18] Emrul Hasan, Mizanur Rahman, Chen Ding, Jimmy Xiangji Huang, Shaina Raza, "Review-based recommender systems: A survey of approaches, challenges and future perspectives," unpublished (arXiv), 2024.

[19] Marko Harasic, Felix-Sebastian Keese, Denny Mattern, Adrian Paschke, "Recent advances and future challenges in federated recommender systems: A survey," International Journal of Data Science and Analytics , vol. 17, pp. 337–357, 2024.

[20] L. Wang, H. Zhou, Y. Bao, X. Yan, G. Shen, X. Kong, "Horizontal federated recommender system: A survey," ACM Computing Surveys , 2024.

[21] Y. Liu, et al. , "Federated recommender systems based on deep learning," Expert Systems with Applications , 2024.

[22] Z. Cai, T. Tang, S. Yu, Y. Xiao, F. Xia, "Marking the pace: A blockchain-enhanced privacy-traceable strategy for federated recommender systems," unpublished (arXiv), 2024.

[23] C. Zhang, G. Long, Z. Zhang, Z. Li, B. Yang, "Personalized recommendation models in federated settings: A survey," unpublished (arXiv), 2025.

[24] W. Yuan, L. Qu, L. Cui, Y. Tong, X. Zhou, H. Yin, "HeteFedRec: Federated recommender systems with model heterogeneity," unpublished (arXiv), 2023.

[25] Y. An, et al. , "Recommender system: A comprehensive overview of concepts, techniques, and challenges," in Proceedings of TSCC 2024, ICCK , 2024.

[26] Yunqi Mi, Jiakui Shen, Guoshuai Zhao, Jialie Shen, Xueming Qian, "A scenario-oriented survey of federated recommender systems: Techniques, challenges, and future directions," unpublished (arXiv), 2025.

[27] Y. Koren, "Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model," in Proc. 14th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining (KDD'08), 2008, pp. 426–434.

[28] Y. Koren and R. Bell, "Advances in Collaborative Filtering," in Recommender Systems Handbook, F. Ricci, L. Rokach, B. Shapira, and P. Kantor, Eds., Springer, 2011, pp. 145–186.

[29] S. Al-Anazi, et al., "Collaborative Filtering Recommendation Algorithm Based on User Correlation and Evolutionary Clustering," Complex & Intelligent Systems, vol. 6, no. 1, pp. 147–156, 2020.

[30] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proc. 10th Int. World Wide Web Conf. (WWW '01), 2001, pp. 285–295.

[31] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, Jan./Feb. 2003.

[32] P. Boström and T. Björkegren, "Comparison of user based and item based collaborative filtering recommendation services," M.Sc. thesis, Uppsala University, 2017.

[33] X. Su and T. M. Khoshgoftaar, "A Survey of Collaborative Filtering Techniques," Adv. in Artificial Intelligence, 2009.