Hybrid Movie Recommendation System with Content-Based and Memory-Based Collaborative Filtering based on Deep Neural Network

Yongmao Yang and **Kampol Woradit**[†], Non-members

ABSTRACT

Recommendation systems assist users in filtering qualified data from vast datasets. Content-based and collaborative filtering techniques are the most widely used among the various recommendation models. In recent years, neural network algorithms have been employed to tackle recommendation problems. However, while these methods reduce the error between true and predicted values, personalized recommendations remain relatively lacking. To address this, we propose a movie recommendation system based on deep neural networks and user vocabulary preference features to alleviate cold start and sparsity issues, reduce prediction errors, and improve recommendation efficiency. We evaluate our model using hit rate (*HR*) and average reciprocal hit rank (*ARHR*) as indicators, achieving an *HR* of 0.76 and an *ARHR* of 0.38. The robustness of our model is demonstrated through comparisons with other studies.

Keywords: Recommendation system, Content-based, Collaborative filtering, Movie, Cold-start, Sparsity, Personalized, Deep neural network

1. INTRODUCTION

Since the advent of the Internet, data has increased exponentially across various domains, rendering it impractical for individuals to manually filter and identify their preferred items from the vast volumes of information. In the early 1990s, several recommendation systems (RSs) were introduced to address this challenge, primarily leveraging content-based filtering (*CBF*) and collaborative filtering (*CF*) algorithms. A diverse array of recommendation algorithms has since been developed upon these foundational methods.

The field has persistently grappled with two major issues: the cold start problem and data sparsity. Various strategies have been proposed to alleviate these challenges and enhance the accuracy of prediction outcomes. The cold start problem arises due to the lack of configuration files for new users or new items, preventing effective recommendations. Data sparsity, on the other hand, occurs because the data is too sparse to calculate all similarities, resulting in missed optimal similarity recommendations.

In the context of movie recommendation systems, user ratings, demographics (e.g., age, gender), comments, and movie texts are employed in *CBF* and *CF* approaches to mitigate the cold start and sparsity issues.

In [1], Ni et al. proposed a collaborative filtering (*CF*) model based on Term Frequency-Inverse Document Frequency (*TF-IDF*) and user characteristics. The model extracts features from users' movie rating data and *TF-IDF*, incorporating users' demographics (e.g., age, gender) to enhance performance. By fully considering the impact of popular items and user characteristics on recommendation results, they mitigated the user cold-start problem. However, the movie cold-start issue remains unresolved, and the similarity between two similar users may be very low due to the sparsity problem.

In [2], Mngomezulu et al. proposed a content-based collaborative filtering movie recommendation system using keyword extraction. The content-based filtering (*CBF*) in their paper is not based on item comparisons; instead, Term Frequency-Inverse Document Frequency (*TF-IDF*) and rapid automatic keyword extraction (*RAKE*) are used to obtain movie plot features for recommendations. This approach reduces runtime in movie recommendation systems and addresses the sparsity problem. However, the model does not consider the user cold-start problem. Additionally, with *TF-IDF*, common words may appear frequently in long documents, resulting in higher *TF* values for these words. Even if they are not important in the document set, they will receive higher weights. This bias can affect the accuracy of keyword extraction.

In [3], Priyati et al. studied four matrix decomposition algorithms based on the MovieLens, Jester, and BookCrossing databases, comparing their root mean square error (*RMSE*) and running time. The results show that alternating least squares (*ALS*) performs best in terms of prediction accuracy and runtime.

In [4], Reddy et al. proposed a content-based movie recommendation system using genre correlation. The recommendation system (*RS*) is built on the types of genres that users might prefer to watch. The approach adopted is content-based filtering using genre correlation. The model only makes recommendations

Manuscript received on July 24, 2024; revised on August 24, 2024; accepted on September 23, 2024. This paper was recommended by Associate Editor Siraporn Sakphrom.

The authors are with Department of Computer Engineering, Faculty of Engineering, Chiang Mai University, Thailand.

[†]Corresponding author: kampol.w@cmu.ac.th

©2025 Author(s). This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License. To view a copy of this license visit: https://creativecommons.org/licenses/by-nc-nd/4.0/.

Digital Object Identifier: 10.37936/ecti-eec.2525231.255176

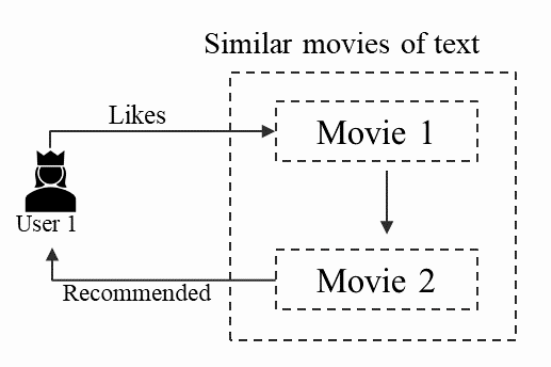

***Fig. 1**: Theory of content-based movie recommendation.*

based on genres, which may affect the accuracy of the recommendations. Additionally, the model does not consider that users may give a low rating to a movie that includes certain genres.

We proposed a novel movie recommendation system (*RS*) model incorporating user word features and genre features. We built a user word list based on ratings and movie text features. Additionally, user genre preferences are calculated using movie genres and ratings to build a deep neural network (*DNN*) model. This model captures the non-orthogonal relationships between users and movies concerning genres.

The research contributions of this paper are as follows.

- This paper addresses and solves all sparsity and cold-start (new user/movie) problems.
- Personalized recommendations are generated by creating user word preference feature vectors.
- The nonlinear relationship between user genre preferences and movie genres is effectively captured through multi-layer nonlinear transformations.
- There is no need to build content similarities between users and movies, improving the efficiency of recommendations and fully considering users' preferences for movies.
- The robustness of user/movie similarity measurement is enhanced by assigning specific weights to multiple similarity metrics and combining them into a single, weighted similarity score.

The organization of the paper is as follows. Section II describes the existing methods. Section III describes the proposed method. Section IV presents the results and discussions, and Section V concludes the paper.

2. EXISTING METHODS

This paper will utilize some existing recommendation system models and algorithms. This section introduces these models and algorithms in detail, including content-based filtering (*CBF*), memory-based collaborative filtering, log-likelihood (*LL*), Term Frequency-Inverse Document Frequency (*TF-IDF*), similarity measures, and deep neural networks (*DNN*).

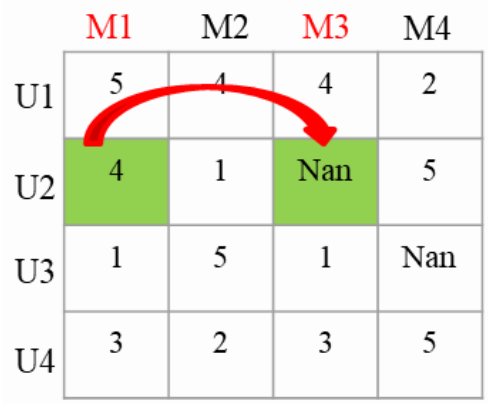

***Fig. 2**: Theory of IBCF movie recommendation.*

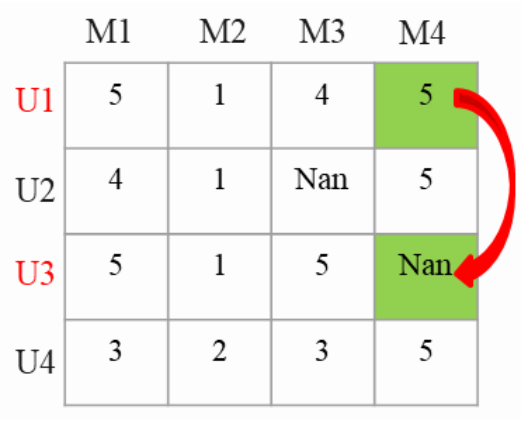

***Fig. 3**: Theory of UBCF movie recommendation.*

2.1 CBF

The principle of content-based filtering (*CBF*) is to predict other movies that users may be interested in and recommend them based on user behavior data and movie content characteristics. The recommendation process of the *CBF* model is illustrated in Fig. 1. For instance, if Movie 1 and Movie 2 are similar in terms of text content and User 1 likes Movie 1, then Movie 2 can be recommended to User 1.

The model can generate recommendations for new movies (addressing the movie cold-start problem) [4]. Additionally, it requires only movie content information, rather than extensive user behavior data. However, content-based filtering (*CBF*) cannot account for novel content or new movie genres that users have not previously shown interest in [5].

2.2 Memory-based CF

Memory-based collaborative filtering (*CF*) is a component of collaborative filtering and includes both item-based collaborative filtering (*IBCF*) and user-based collaborative filtering (*UBCF*).

The basic principle of *IBCF* is that if a user is interested in a movie, they may also be interested in other movies similar to it. Fig. 2 illustrates an example of *IBCF* in

***Table 1**: Contingency table for word frequencies.*

	CORPUS 1	CORPUS 2	TOTAL
Freq of word	a	b	a+b
Freq of other word	c-a	d-b	c+d-a-b
TOTAL	c	d	c+d

movie recommendation systems. In this example, M1 and *M3* are similar movies based on ratings, so the *NaN* value for M3 in the rating matrix for User 2 (*U2*) can be filled with the number 4.

The basic principle of *UBCF* is that if a user has similar interests and behaviors to another user, the former is likely to be interested in items that the latter likes. Fig. 3 is an example diagram of *UBCF*. Where *U1* and *U3* are similar users based on ratings, the *NaN* value of M1 for *U3* can be filled with the number 5[6].

Memory-based collaborative filtering (*CF*) can be updated in real time as new data is added, without the need to retrain the model. However, sparsity can lead to insufficient information when calculating similarity, which may affect the effectiveness of the recommendations.

2.3 Log-likelihood Calculation

Log-likelihood (*LL*) calculates the importance of each word in *corpus 1* relative to *corpus 2* through a log-likelihood measure. This paper will use *LL* instead of Term Frequency-Inverse Document Frequency (*TF-IDF*) to determine the importance of each word and utilize it as the feature vector for the movie [7].

The calculation formula is (1) and (2). The formula of *LL* is given by

$$E_i = \frac{N_i * \sum_i O_i}{\sum_i N_i} \tag{1}$$

$$LL = 2 * \sum_i O_i * \ln \frac{O_i}{E_i} \tag{2}$$

In Table 1, let $N1 = c$, and $N2 = d$. For this word a, the expected counts are $E1 = c * (a + b)/(c + d)$ and $E2 = d * (a + b)/(c + d)$. The log-likelihood (LL) is calculated as $LL = 2 * ((a * \log(a/E1)) + (b * \log(b/E2)))$ [7].

2.4 TF-IDF

Term Frequency-Inverse Document Frequency (*TF-IDF*) calculates the importance of a word in a document based on its frequency and prevalence across a collection of documents. The formula is given by:

$$TF_{i,d} = \frac{f_{i,d}}{\sum_k f_{k,d}}, \tag{3}$$

$$IDF_{i,D} = \log(\frac{N}{|\{d \in D : i \in d\}|}), \tag{4}$$

$$TF - IDF_{i,d,D} = TF_{i,d} * \ IDF_{i,D}, \tag{5}$$

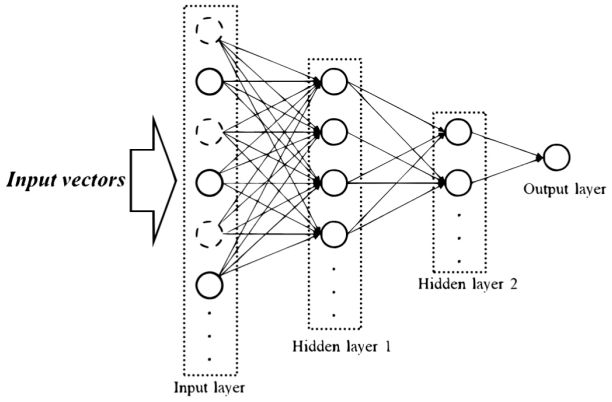

***Fig. 4**: DNN network architecture.*

where $TF_{i,d}$ is the frequency of word i in document d, and $IDF_{i,D}$ denotes the prevalence of word i across the entire document collection. The *TF-IDF* will be used to calculate each word's relative importance to users [8, 9].

2.5 Similarity Measure

This paper will use cosine similarity. Since the cosine similarity measure is not affected by the length of the vectors, it can reduce calculation errors when comparing vectors of different sizes [10]. The formula for cosine similarity is given by (6).

$$\cos_{a,b} = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}|| \, |\boldsymbol{b}||} \tag{6}$$

2.6 DNN

The neural network (*NN*) model offers the highest recommendation efficiency. However, it has drawbacks including long training times, numerous parameters to adjust, high complexity, and the need to process *NaN* values in the input data. Neural networks primarily consist of an input layer, hidden layers, and an output layer. Currently, there are many types of *NN* models, such as deep neural networks (*DNN*), convolutional neural networks (*CNN*), and recurrent neural networks (*RNN*). In the proposed model, the *DNN* will capture the nonlinear relationship between user genre preferences and movie genres. Fig.4 illustrates the architecture of the *DNN* [11].

3. PROPOSED METHOD

The approach to building the recommendation system involves hybrid filtering using content-based filtering (*CBF*) and collaborative filtering (*CF*). As discussed earlier, the user's preference for text content and genres can be determined by combining the text content and genre characteristics of movies, based on the user's ratings.

The framework of the proposed method is shown in Fig.5 . It is divided into 10 main steps:

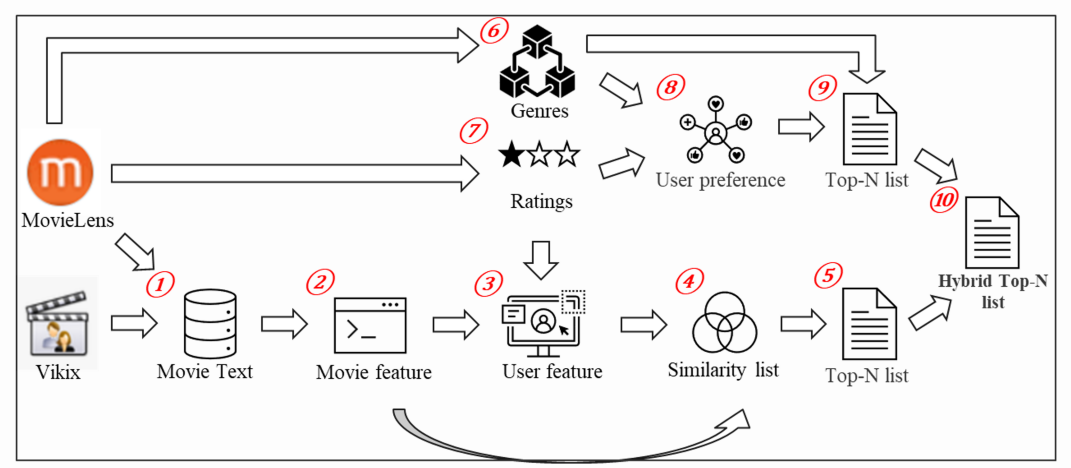

***Fig. 5**: Variation of spacing with suspension preload.*

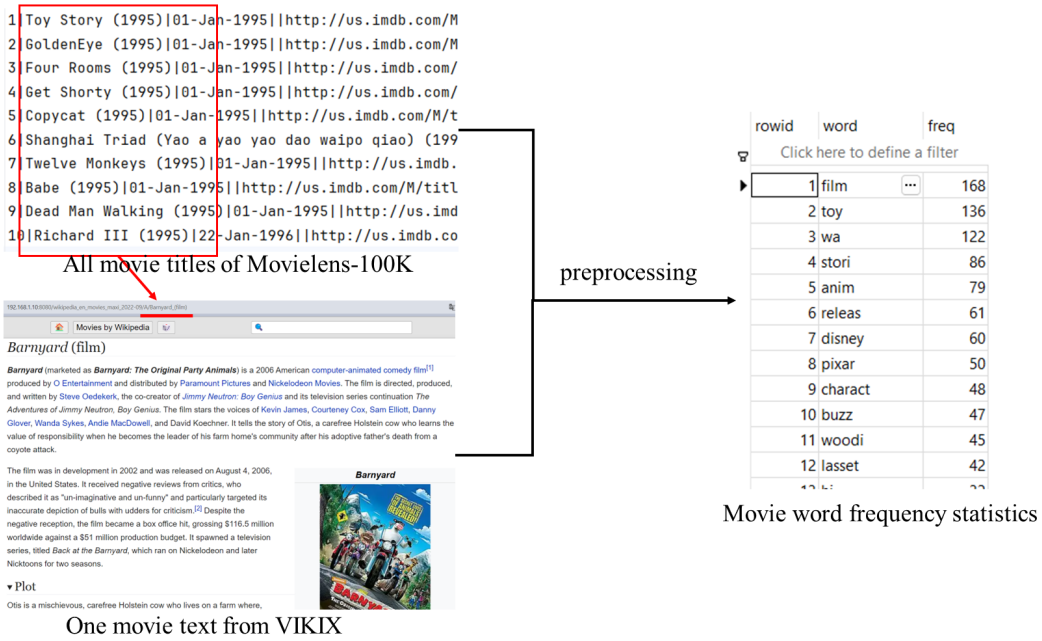

***Fig. 6**: The detail process of step 1.*

Step 1: Obtain movie texts from sources such as movie titles and the Wikipedia Movie Database, including background information, plot summaries, awards, etc. Count the word frequencies in the texts.

Step 2: Calculate movie feature vectors using the log-likelihood method. Each word will have an LL value representing its importance.

Step 3: Generate user feature vectors based on movie features and ratings using the *TF-IDF* method.

Step 4: Use the cosine similarity method to calculate the similarity between users and movies.

Step 5: Generate a Top-N list after selecting the threshold for the recommendation.

Step 6: Obtain genre features for each movie, including categories such as action, horror, animation, comedy, etc.

Step 7: Record the user's Likert rating for the movie (on a scale of 1-5).

Step 8: Use deep neural networks (*DNN*) to model non-orthogonal relationships between users and movies. Recommend new users (addressing the user cold-start problem) based on their genre preference ratings.

Step 9: After selecting a threshold, obtain a list of the Top-N movies with the highest recommendation scores.

words	LL
Click here to define a filter	
toy	1006.4503482411
pixar	429.422590890739
buzz	373.896331807901
lasset	356.924281621896
woodi	306.438123475846
anim	270.539923403626
stori	217.533627292411
disney	199.774518379761
lightyear	139.548524320963
comput	113.424033036703
andi	107.66265321526
katzenberg	106.010671492657
charact	61.6533827664195
whedon	61.2876246951691
docter	54.6379551728821
rc	54.6379551728821

***Fig. 7**: Movie 1: Feature vector is made of LL.*

***Fig. 8**: User 1: word cloud.*

Step 10: Combine the movies from both recommendation lists to create the final recommended list.

Some details of the steps need to be specified here.

In **Step 1**, movie titles will be obtained from the *ML-100k* dataset and used to generate URL links to *VIKIK*. Subsequently, each movie's text can be retrieved using *PyCharm* with Python. The process is illustrated in Fig. 6.

At the end of **Step 1**, we will have the word frequency statistics for each movie.

In **Step 2**, each movie feature vector will be calculated using the log-likelihood (*LL*) method, as described by (1) and (2). Fig.7 illustrates the content feature vector with *LL* for Movie 1.

In **Step 3**, when User 1 rates some movies, the words from these movies are extracted. If a word exists in the feature vector of a movie, it is assigned a value of 1; otherwise, it is assigned a value of 0. This process generates a word frequency vector for User 1. *TF-IDF* is then used to create the user's feature vector based on these words. Fig.8 shows the *TF-IDF* word cloud for User 1. In this cloud, larger word sizes indicate a greater preference by the user for movies related to those words.

In **Steps 4 and 5**, both movie and user features are represented by words, allowing us to compute the cosine similarity (*COS*) between movies and users. The *COS*

	Action	Adventure	Animation	...	Thriller	War	Western
1	0.711538	0.500000	0.666667	...	0.904762	0.909091	1.0
2	1.000000	1.000000	1.000000	...	0.900000	0.500000	0.5
3	0.375000	1.000000	0.500000	...	0.250000	0.750000	0.5
4	0.900000	0.750000	0.500000	...	0.900000	1.000000	0.5
5	0.657895	0.666667	0.875000	...	0.625000	0.700000	1.0
..	...	...	...	...	...	...	...
939	0.833333	0.700000	1.000000	...	1.000000	1.000000	0.5
940	0.777778	0.625000	1.000000	...	0.785714	0.700000	0.5
941	0.800000	0.666667	1.000000	...	0.750000	1.000000	0.5
942	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.0
943	0.833333	0.750000	0.500000	...	0.892857	0.812500	0.5

***Fig. 9**: Ratings of users' movie genre preferences.*

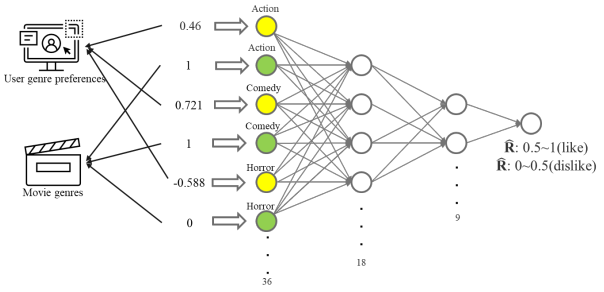

***Fig. 10**: Deep neural network construction framework.*

value ranges from 0 to 1 and indicates the likelihood of recommending a movie to the user. To generate the recommendation list, we select a threshold for the *COS* values, using the Top-N method to filter movies that the user may prefer.

In **Steps 6 and 7**, to obtain the movie genre and rating, we use Python to generate a binary representation where the presence of a genre is indicated by 1 and its absence by 0. The genres considered include action, adventure, animation, children, comedy, and others, encompassing a total of 18 genres.

In **Steps 8 and 9**, we define that a user likes a movie if its rating exceeds 3. The model then classifies the rated movies into two categories: a "liking" list and a "disliking" list. The user's genre preferences are subsequently computed using (7). (see Fig. 9)

$$\hat{R}g = \frac{N_{likes} - N_{dislikes}}{Max(N_{likes} - N_{dislikes})} \tag{7}$$

In the *DNN* model, we use Binary Cross-Entropy (*BCE*) as the loss function to handle binary classification (like or dislike) problems. The *BCE* function is given by (8).

$$BCE = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log \hat{y}_i + (1 - y_i) \log\left((1 - \hat{y}_i)\right)\right] \tag{8}$$

The process of the *DNN* model is illustrated in Fig. 10. The output of the model ranges from 0 to 1; the closer the result is to 1, the higher the recommendation index. Adaptive Moment Estimation (Adam), which features an adaptive learning rate and strong robustness, will be used to train the sub-model. The learning rate is set to 0.001, the activation function used is *ReLU*, and the network

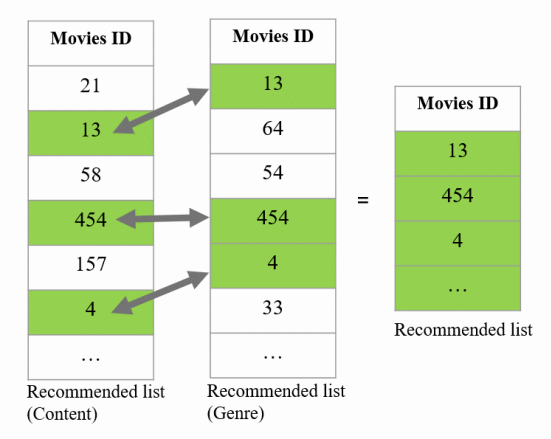

***Fig. 11**: User 1: Recommendation list display.*

***Table 2**: Contingency table for word frequencies.*

The recommendation list of Top-10	
Movie ID	17
	313
	42
	221
	156
	658
	31
	323
	4
	176

architecture includes an input layer with 32 neurons, hidden layers with 18 and 9 neurons respectively, and an output layer with 1 neuron. After training, the Top-N method is applied to filter movies that the user may like.

In **Step 10**, the two recommendation lists obtained from the previous steps are combined to enhance recommendation performance. We extract the common movies from both lists and use them as the final recommendation result. The final recommendation list is shown in Fig. 11.

Steps 1 to 5 utilize traditional content-based recommendation methods, while steps 6 to 9 employ collaborative filtering with deep neural networks. Therefore, the hybrid recommendation system model proposed in this paper integrates content-based and collaborative filtering algorithms. When a new user provides a rating for a movie category, steps 6 to 9 help mitigate the cold start

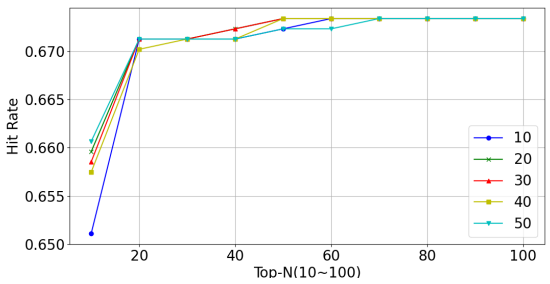

***Fig. 12**: Hit rate evaluation about ML-100K.*

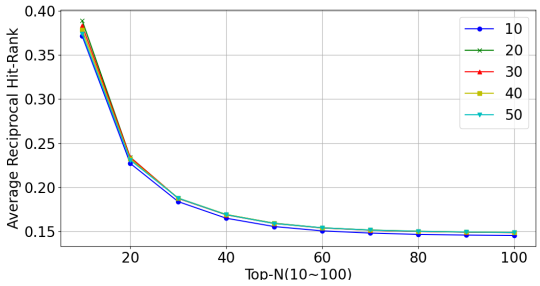

***Fig. 13**: Average reciprocal hit-rank evaluation about ML-100K.*

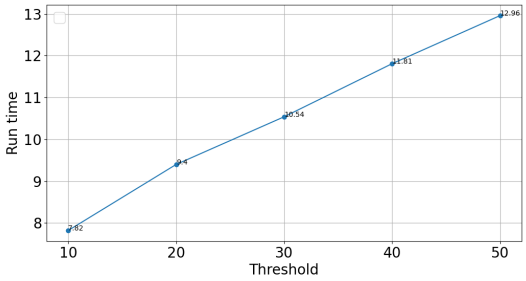

***Fig. 14**: Impact of threshold size on running time.*

problem for new users and new movies, enabling the system to generate a recommendation list that includes both new and existing movies. Table 2 shows the top 10 movies in the list generated after simulating a new user randomly rating various movie genres.

4. RESULTS AND DISCUSSIONS

4.1 Dataset

The proposed model will be evaluated using the MovieLens dataset, which will be divided into a training set and a test set. The details of the dataset used are shown in Table 3.

4.2 Evaluation Metrics

The recommender list of the recommendation system (*RS*) will use the Top-N method, making the accuracy and order of recommendations critical. To evaluate the recommendation performance of the model, we will use Hit Rate (*HR*) and Average Reciprocal Hit Rank (*ARHR*). *HR* is commonly employed to assess the system's effectiveness and indicates the proportion of recommended items that the user is genuinely interested in. Specifically, M is the total number of test users, and N is the number of users whose movies are in the test set's recommendation list [12], the hit rate can be calculated as shown in (9).

$$HR = \frac{M}{N} \tag{9}$$

ARHR is an indicator used to evaluate the quality of recommendations in recommendation systems. It

***Table 3**: Details for each dataset.*

	MovieLens(ML-100K)	**VIKIX**
Users	943	-
Movies	1682	-
Ratings	100,000	-
Genres	18	-
Texts	-	300,000+
Links	https://grouplens.org/datasets/movielens/100k/	https://library.kiwix.org

***Table 4**: Hit rate experimental data.*

	$N_k = 10$	$N_k = 20$	$N_k = 30$	$N_k = 40$	$N_k = 50$	$N_k = 60$	$N_k = 70$	$N_k = 80$	$N_k = 90$	$N_k = 100$
$N_t = 10$	0.651	0.671	0.671	0.671	0.672	**0.673**	0.673	0.673	0.673	0.673
$N_t = 20$	0.659	0.671	0.671	0.672	**0.673**	0.673	0.673	0.673	0.673	0.673
$N_t = 30$	0.658	0.671	0.671	0.672	**0.673**	0.673	0.673	0.673	0.673	0.673
$N_t = 40$	0.657	0.670	0.671	0.671	**0.673**	0.673	0.673	0.673	0.673	0.673
$N_t = 50$	0.660	0.671	0.671	0.671	0.672	0.672	**0.673**	0.673	0.673	0.673

***Table 5**: Average reciprocal hit-rank experimental data.*

	$N_k = 10$	$N_k = 20$	$N_k = 30$	$N_k = 40$	$N_k = 50$	$N_k = 60$	$N_k = 70$	$N_k = 80$	$N_k = 90$	$N_k = 100$
$N_t = 10$	**0.371**	0.227	0.183	0.164	0.155	0.150	0.148	0.146	0.145	0.145
$N_t = 20$	**0.388**	0.234	0.187	0.169	0.159	0.154	0.151	0.150	0.149	0.148
$N_t = 30$	**0.383**	0.234	0.187	0.168	0.158	0.153	0.151	0.149	0.149	0.148
$N_t = 40$	**0.378**	0.232	0.187	0.169	0.159	0.154	0.151	0.150	0.149	0.148
$N_t = 50$	**0.374**	0.230	0.187	0.168	0.158	0.154	0.151	0.150	0.149	0.148

calculates the average of the reciprocal rankings of hit items in the recommendation list, providing a more detailed reflection of the system's performance [13], and is given by

$$ARHR = \frac{1}{|U|} \sum_{u \in U} \sum_{i \in L_u} \frac{1}{rank_{u,i}} \tag{10}$$

where:

- $\boldsymbol{L_u}$ is the set of items liked or selected by user **u**.
- $\boldsymbol{rank_{u,i}}$ is the ranking position of item i in the user u's recommendation list.
- $|\boldsymbol{U}|$ is the total number of users.

4.3 Result

We will use *HR* and *ARHR* to evaluate our proposed model, To ensure an accurate evaluation, we need to determine the threshold (N_t) for the number of words in the movie features. Additionally, we must set the number of top recommendations (N_k). We will train the model with varying N_t and N_k values, where N_t belongs to $\{10, 20, 30, 40, 50\}$, and N_k belongs to $\{10, 20, 30, \ldots, 100\}$. The final evaluation results are presented in Fig. 12 (*HR*) and Fig. 13 (*ARHR*). Tables 4 and 5 provide detailed information corresponding to Figs. 12 and 13. The training results indicate that the optimal *HR* is 0.673 and the optimal *ARHR* is 0.388. Fig. 14 illustrates the running time for different threshold sizes. The threshold size has minimal impact on recommendation efficiency. However, a lower threshold can reduce the running time of the program.

***Table 6**: Details for each dataset.*

Models	**References**	**HR@10**	**ARHR@10**
SAR	[14]	0.341	0.346
ALS	[15]	0.476	0.257
NCF	[16]	0.329	0.335
BPR	[17]	0.389	0.340
BIVAE	[18]	0.415	0.341
LightGCN	[19]	0.364	0.347
Proposed CBF		0.452	0.228
Proposed CF		0.487	0.249
Proposed hybrid model		**0.671**	**0.388**

4.4 Discussions

To verify the robustness of the proposed model, we compared it with other existing models using the *HR* and *ARHR* evaluation index. The comparison results, as shown in Table 6, indicate that the proposed model outperforms the others, where the best *HR* and *ARHR* achieved by the content-based sub-model and the deep neural network sub-model are (0.452, 0.22), and (0.487, 0.24), respectively. Experimental results demonstrate that the hybrid model outperforms each individual approach (see Table 6). It is noteworthy that the *HR* reported in this paper is significantly higher than those reported in other studies. This improvement is due to the use of novel recommendation techniques in both sub-models. Additionally, the deep neural network model further enhances performance by effectively selecting and optimizing features based on users' textual preferences for movies.

An analysis of the evaluation results for *HR* and *ARHR* reveals that *HR* increases as N_k increases and gradually stabilizes after N_k exceeds 50. This trend occurs because,

with a larger recommendation list, there is a higher probability of including relevant movies, which naturally leads to an increase in *HR*. However, as the list size grows, the marginal benefit of adding additional movies diminishes, causing *HR* to stabilize. Conversely, *ARHR* decreases rapidly as N_k increases and gradually stabilizes after N_k exceeds 50. This decline occurs because as more movies are included in the recommendation list, relevant movies are more likely to be ranked lower, reducing their reciprocal rank value and thus lowering *ARHR*. After N_k reaches 50, the impact of adding more movies becomes negligible, leading to the stabilization of *ARHR*. This stabilization indicates that the list size has reached a point where further increases have minimal effect on the ranking positions of relevant movies, causing the *ARHR* value to level off.

5. CONCLUSION

We propose a hybrid recommendation model that combines content-based (*CB*) and collaborative filtering (*CF*) approaches. The collaborative filtering deep neural network sub-model in this hybrid recommendation system can effectively address the cold start problem for new users and movies. It achieves this by calculating a user's preference for a movie based on both the input ratings of the user for the movie category and the category of the movie, applicable to both new/old users and movies, and significantly alleviates the sparsity issue. In the content-based filtering (*CBF*) submodel, unlike traditional content-based algorithms, our approach leverages the textual features of all movies rather than relying solely on the similarity of user preferences for generating recommendations. The user features derived from the movie text can be utilized to clarify the user's detailed preference characteristics, which is more effective in determining whether a new movie is suitable for recommendation.

Additionally, the results indicate that while the choice of threshold has a minimal effect on recommendation quality, a larger threshold results in increased program running time. Therefore, to enhance the efficiency of the recommendation system, it is advisable to select a lower threshold.

REFERENCES

[1] J. Ni, Y. Cai, G. Tang, and Y. Xie, "Collaborative filtering recommendation algorithm based on tf-idf and user characteristics," *Applied Sciences*, vol. 11, no. 20, p. 9554, 2021.

[2] M. Mngomezulu and R. Ajoodha, "A contentbased collaborative filtering movie recommendation system using keywords extractions," in *2022 International Conference on Engineering and Emerging Technologies (ICEET). IEEE*, 2022, pp. 1–6.

[3] Priyati, A. D. Laksito, and H. Sismoro, "The comparison study of matrix factorization on collaborative filtering recommender system," in *2022 5th International Conference on Information and Communications Technology (ICOIACT). IEEE*, 2022, pp. 177–182.

[4] S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok, and B. Venkatesh, "Content-based movie recommendation system using genre correlation," in *Smart Intelligent Computing and Applications: Hybrid Movie Recommendation System with Content-Based and Memory-Based Collaborative Filtering based on Deep Neural Network9 Proceedings of the Second International Conference on SCI 2018*, Volume 2. Springer, 2019, pp. 391–397.

[5] K. A. Reshak, B. N. Dhannoon, and Z. N. Sultani, "Hybrid recommender system based on matrix factorization," in *AIP Conference Proceedings*, vol. 2457, no. 1. AIP Publishing, 2023.

[6] P. Thakkar, K. Varma, V. Ukani, S. Mankad, and S. Tanwar, "Combining user-based and itembased collaborative filtering using machine learning," in *Information and Communication Technology for Intelligent Systems: Proceedings of ICTIS 2018*, Volume 2. Springer, 2019, pp. 173– 180.

[7] K. Cosh, "Current research themes in software engineering: An application of text mining," in *2016 8th International Conference on Knowledge and Smart Technology (KST). IEEE*, 2016, pp. 125–129.

[8] M. R. Hasan and J. Ferdous, "Dominance of ai and machine learning techniques in hybrid movie recommendation system applying textto-number conversion and cosine similarity approaches," *Journal of Computer Science and Technology Studies*, vol. 6, no. 1, pp. 94–102, 2024.

[9] H. Kumaar, S. Srikumaran, S. Veni et al., "Content-based movie recommender system using keywords and plot overview," in *2022 International Conference on Wireless Communications Signal Processing and Networking (WiSPNET). IEEE*, 2022, pp. 49–53.

[10] S. Airen and J. Agrawal, "Movie recommender system using k-nearest neighbors variants," *National Academy Science Letters*, vol. 45, no. 1, pp. 75–82, 2022.

[11] M. F. Aljunid and M. Dh, "An efficient deep learning approach for collaborative filtering recommender system," *Procedia Computer Science*, vol. 171, pp. 829–836, 2020.

[12] M. Moradi and J. Hamidzadeh, "Ensemble-based top-k recommender system considering incomplete data," *Journal of AI and Data Mining*, vol. 7, no. 3, pp. 393–402, 2019.

[13] Jadon and A. Patil, "A comprehensive survey of evaluation techniques for recommendation systems," *arXiv preprint arXiv:2312.16015*, 2023.

[14] S. Gong, "A collaborative filtering recommendation algorithm based on user clustering and item clustering." *J. Softw.*, vol. 5, no. 7, pp. 745–752, 2010.

[15] Priyati, A. D. Laksito, and H. Sismoro, "The comparison study of matrix factorization on collaborative filtering recommender system," in *2022 5th*

International Conference on Information and Communications Technology (ICOIACT). IEEE, 2022, pp. 177–182.

[16] L. Sang, M. Xu, S. Qian, and X. Wu, "Knowledge graph enhanced neural collaborative recommendation," *Expert systems with applications*, vol. 164, p. 113992, 2021.

[17] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[18] Q.-T. Truong, A. Salah, and H. W. Lauw, "Bilateral variational autoencoder for collaborative filtering," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 292–300.

[19] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 639– 648.

Yang Yongmao born in 1998, he obtained a bachelor's degree in computer electronics and technology in Nanjing, China in 2020, and is currently studying as a graduate student in computer engineering at Chiang Mai University, Thailand, Research interests are machine learning, recommender systems, and natural language processing.

Kampol Woradit (Member, IEEE) received the B.Eng. (Hons.) and Ph.D. degrees in electrical engineering from Chulalongkorn University, Thailand, in 2002 and 2010, respectively. He was a Visiting Student with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, from 2007 to 2008, and was on a Postgraduate Research Attachment with the Modulation and Coding Department, Institute for Infocomm Research, Singapore. He was a Faculty Member with the Department of Electrical Engineering, Srinakharinwirot University, Thailand, from 2010 to 2020. He is currently a Faculty Member with the Department of Computer Engineering, at Chiang Mai University, Thailand. His research interests include wireless communications, the Internet of Things, and physical layer security. He received the ECTI-CON Best Paper Award in 2019 and the Royal Golden Jubilee Scholarship from the Thailand Research Fund for his Ph.D. Program.