A Novel Time-Aware Hybrid Recommendation Scheme Combining User Feedback and Collaborative Filtering

Hongzhi Li and Dezhi Han

***Abstract*—Nowadays, recommender systems are used widely in various fields to solve the problem of information overload. Collaborative filtering and content-based are representative solutions in recommender systems, however, the content-based model has some shortcomings, such as single kind of recommendation results, lack of effective perception of user preferences; while, for the collaborative filtering model, there is a cold start problem, and such a model is greatly affected by its adopted clustering algorithm. To address these issues, a hybrid recommendation scheme is proposed in this article, which is based on both collaborative filtering and content-based. In this scheme, we propose the concept of time impact factor, and a time-aware user preference model is built based on it. Also, user feedback on recommendation items is utilized to improve the accuracy of our proposed recommendation model. Finally, the proposed hybrid model combines the results of content recommendation and collaborative filtering based on the logistic regression algorithm.**

***Index Terms*—Collaborative filtering (CF), time-aware, time impact factor, user feedback.**

I. Introduction

With the rapid development of the Internet, users can enjoy rich information services and convenient social interaction through Internet applications [1]. However, the information overload problem in Internet applications is becoming more and more serious, which makes it difficult for users to choose what they really like. Therefore, various recommendation models are widely used to help users locate information. In general, these popular recommendation models can be divided into collaborative filtering (CF), content-based, and hybrid approaches. The CF method [2]–[4] is based on the view that the higher the similarity between users, the more overlapping of user preferences. The content-based approach [5], [6] is based on representations to recommend items and these representations are usually extracted from descriptions. It is necessary to calculate the similarity between item representations and user profiles. The hybrid approach [9] generates recommendations by combining several other methods. This approach is based on the idea that the hybrid method should take advantage of other approaches and avoid the disadvantages of each approach to achieve better recommendations.

Among these approaches, CF is used widely in the field of e-commerce. Content-based performs well in recommending blogs, news, and documents. Generally, the CF has a better performance than the content-based model. Note that the better performance is based on sufficient user information, including personal information and behavior information. Moreover, the CF model usually suffers from the cold start problem due to a lack of adequate rating records [3], [5], in which case the content-based model seems to be an alternative approach. However, this approach has its limitations. For instance, users' profiles cannot be accurately acquired because of a lack of sufficient user behavior information [2], [6]. Another issue is that the content-based approach is slow to perceive the change of user preference. In fact, the user's interest usually changes with time. For instance, a fashion windbreaker suitable for autumn is launched. A user just has such a demand and pay attention to it for a while. But after autumn, he may no longer need such clothes, and the relevant behavior records are remained in the recommendation model and affect the recommendation results. This is because that the content-based model only depends on users' preferences for certain items in the past, the recommendations generated by content-based (CB) will be similar to those that users used to like.

The hybrid recommendation model provides a new approach to solve the above issues. Different kinds of hybrid recommendation models have been proposed for Internet applications [7], [8], such as weighting results of different recommended techniques, using a switching mechanism (i.e., this mechanism changes and adopts different recommendation technologies according to the background and actual situations of problems). However, there is no research considering the impact of time on user preferences in the hybrid model. Based on our experience, the time factor has a significant impact on user preferences, and these preferences are usually changing with time. Thus, we should pay attention to the time factor that affects user preferences. Moreover, we can improve the recommender system by utilizing feedback from users. Therefore, we propose a novel hybrid recommendation model, which contains three key points.

1) For building user profiles, we define the time factor as a weighted basis for selecting behavior records. In this selection process, we will focus on the generated behavior records that

Manuscript received May 9, 2020; revised September 6, 2020; accepted October 5, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61873160 and 61672338. *(Corresponding author: Dezhi Han.)*

The authors are with the College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China (e-mail: 1071260932@qq.com; 17855009630@163.com).

Digital Object Identifier 10.1109/JSYST.2020.3030035

are closer to the current time. Thus, points of interest that are contained in the selected records should be given priority.

2) A feedback mechanism should be introduced into the model, which can be used to establish feedback libraries based on the user's feedback records (e.g., click rate, browsing duration, etc.) of previous recommendations. Furthermore, the recommendations of our hybrid model should be filtered by the feedback mechanism to improve the accuracy of our model.

3) Spectral clustering algorithm is used to improve the efficiency of CF.

Finally, we use the logistic regression method to aggregate the recommendations from content-based and CF. The remainder of this article is organized as follows. In Section II, we briefly present the related work. In Section III, we provide the background necessary to understand the proposed scheme, such as the preference representation and the spectral clustering algorithm. In Section IV, we describe the proposed scheme in detail, including the definition of time impact factor. In Section V, we introduce the experimental environment and analyze the experimental results. Finally, Section VI concludes the article.

II. Related Work

The main purpose of recommendation models is to provide helpful and suitable items for users. The traditional recommendation approaches, including CF, content-based, and knowledge-based. Specifically, these approaches mainly focus on the fields of online news, social media, online advertising, and e-commerce. However, a single recommendation approach may not perform well, and it is difficult to collect detailed enough user behavior records for privacy concerns. Therefore, more and more attention has been paid to the hybrid approach and a lot of studies have been carried out, recently.

The early hybrid model is mainly used to improve CF. In 2005, Li *et al.* [11] present a hybrid model of CF based on items and users. This model combines both item-based and user-based CF. The similarity calculation between active users and other neighbor users is based on other items related to prediction items, not on all items. Researchers consider introducing various auxiliary information into the recommendation model to build better hybrid models. For instance, the studies in [12]–[14] introduces a hybrid model, which utilizes user-based similarity, point-of-interest (POI)-based similarity, and geographic information to recommend tourist spots. The authors of [15] designed a hybrid trust-based model. This model is applied in the field of online learning, which deals with the issue of data sparsity by incorporating two trust relationships into algorithm computation. To mine more implicit information in hybrid models, the authors of [23] proposed a Bayesian network model combining content-based and CF, and the Bayesian network is used to calculate the joint probability distribution of user access time and resource information to obtain the user's interest of the provided resource. In [17]–[19], the concept of group recommendation is proposed, Boutilier *et al.* [17] developed probabilistic inference methods for predicting individual preferences given observed social connections. Sun *et al.* [19] proposed a social-aware group recommendation framework that jointly utilizes both social relationships and social behaviors to not only infer a group's preference but also model the tolerance and altruism characteristics of group members. In [20], the authors proposed a time-aware hybrid model for topics in microblogs. Since hot topics of microblogging communities change quickly with time, it is necessary to recommend time-sensitive topics. Such a model combines a content-based approach and a time-aware component to find latent topics.

Artificial intelligence technologies provide a new perspective to improve the hybrid model. The authors in [21] proposed a system that uses a sentiment analysis approach to classify user's keywords or ratings as positive and negative. This system will recommend items to users that match their emotional tendencies. In the work of [22] and [24], knowledge graphs are mainly integrated into the recommendation generation process as a dataset with rich semantics. In [25], the authors proposed a hybrid model that builds a graph-based latent factor model. This approach combines the strength of latent factorization with graphs. In [26], the authors proposed a collaborative deep learning model. This model applies the deep neural network and convolution neural network to extract the hidden feature vectors of users and items with sparse ratings to build the rating matrix. Yu *et al.* [30] proposed a multilinear interactive matrix factorization (MF) algorithm (MLIMF) to model the interactions between the users and each event associated with their final decisions. The proposed model considers not only the user-item rating information but also the pairwise interactions based on some empirically supported factors, and this model is used to solve the problem of over-dependence on the user-item rating matrix for MF-based approaches. To solve the issue of the sparse content of items in collaborative retrieval (CR) system, the authors in [32] suggested that the sophisticated relationship of each (query, user, item) triple should be sufficiently explored from the perspective of items. Besides, an alternative factorized model is proposed in [32], which could better evaluate the ranks of those items with sparse information for the given query–user pair.

Overall, the previous research works on the hybrid recommendation mainly focus on solving the sparsity of rating matrix and mining implicit relations between users and items. However, the feedback from users on recommendations and the timeliness of the recommendations have not been paid enough attention. This article is built on the prior work of content-based and CF, and we consider particularly the time factor and user feedback to enhance the performance of recommendation.

III. Preliminaries

A. Preference Representation

User preference should be represented in a way that can be easily processed by computer systems. Usually, natural language descriptions of items should be converted to structures that computers can process directly. To be specific, there are several structures used widely, such as user-item rating matrix [7], user-interests knowledge table [10], keywords vector, vector space model (VSM), and semantic ontology [27]-[29]. In this article, we choose VSM as the user preference structure due to the following reasons.

1) Compared with user-interests knowledge table [10], the structure of the VSM model is relatively simple, and the computation costs of building the VSM model are not very large.

2) Compared with the keywords vector, the VSM model is equivalent to a set of common keywords vector, and it includes more useful information for recommendation computations.

3) Also, the VSM model does not rely on the natural language processing technology, the implementation difficulty of VSM is smaller than that of the semantic ontology method.

After that, we can briefly introduce several concepts about VSM as follows.

1) *Document:* It is usually a fragment with a certain scale in an article, such as sentence, sentence group, paragraph, and paragraph groups. In recommender systems, *documents* mainly refer to items to be recommended (e.g., news, blog, video, music, etc.) or behavior records of users.
2) *Term/Feature Term:* A feature term is the smallest indivisible language unit in VSM, which can be a word, a phrase, and a phrase group. Specific to recommender systems, the term refers to the keywords that can represent the characteristics of recommended items or rating items of users. Therefore, a document can be regarded as a collection of terms and can be expressed as Document $= d(t_1, t_2, \ldots, t_n)$, where t_k ($1 \leq k \leq n$) is a feature term.
3) *Term Weight:* For the document $d(t_1, t_2, \ldots, t_n)$, each term $t_k \in d$ should be assigned a weight w_k to indicate its importance in the document d. Then, such a document d can be expressed as $d(t_1, w_1; t_2, w_2; \ldots; t_n, w_n)$. In recommender systems, the *term weight* represents the weight of feature keywords after the recommended items are converted into vector form or the score of each rated item in user rating vectors.

Therefore, given a document $d(t_1, w_1; t_2, w_2; \ldots; t_n, w_n)$, and conforms the following two principle: 1) Each feature term t_k $(1 \leq k \leq n)$ is different (there is no repetition); 2) there is no sequential relation of each feature term t_k (that is, the internal structure of the document is not considered). Then, we call $d(t_1, w_1; t_2, w_2; \ldots; t_n, w_n)$ as the vector or vector space model of the document. Term frequency-inverse document frequency (TF-IDF) [8] is usually used to calculate the weight of feature terms in VSM. The main idea of TF-IDF is that if a feature term appears frequently in one article, but rarely in other articles, it is considered that this term should be assigned with high weight. We can describe the calculation process of TF-IDF as follows.

Step 1: TF (term frequency) is the number of times a term appears in an article. It can be calculated as

$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{1}$$

where $n_{i,j}$ is number of term$_i$ in document d_j, and the denominator $\sum_k n_{k,j}$ is the total number of all terms in d_j.

Step 2: IDF (inverse document frequency) shows the frequency of a term in all documents. If a term appears in many texts, its IDF should be low. It can be calculated as

$$\text{idf}_i = \log \frac{|D|}{|\{j : t_i \in d_j\}| + 1} \tag{2}$$

where $|D|$ is the total number of all documents, $|\{j : t_i \in d_j\}|$ indicates the number of documents containing the term t_i. We usually use $|\{j : t_i \in d_j\}| + 1$ as the denominator to avoid having a zero denominator.

Step 3: We can calculate the TF-IDF value of the term t_i as $tf_{ij} * \text{idf}_i$, which is shown in (3)

$$\text{TF-IDF}_{(t_i)} = \frac{n_{i,j}}{\sum_k n_{k,j}} * \log \frac{|D|}{|\{j : t_i \in d_j\}| + 1}. \tag{3}$$

After introducing the definition of VSM and the calculation process of TF-IDF in detail, then we can give the definition of preference representation as follows.

Definition 1 (Preference representation): Let $I = \{i_1, i_2, i_3, ..., i_n\}$ represents the user preference that is calculated as VSM, and i_k is the weight of the kth preference.

B. Spectral Clustering

Generally, the similarity between users or items should be calculated in the CF algorithm. For instance, in user-based CF, the first step is to provide a user set composed of k elements with the highest similarity. Then the recommendation of items is based on the interests and preferences of similar users. Consequently, with the increase of users, the efficiency of the recommendation algorithm will decline. To solve this challenge, the clustering algorithm is proposed and applied in the recommendation algorithm. By dividing users into different clusters based on user similarity, the calculations involved are only within the cluster and between different clusters.

The idea of spectral clustering comes from graph theory. The essence is to transform the clustering problem into the optimal partitioning problem of the graph. In the process of spectral clustering, data nodes are used as vertex V in the graph, and edges E between vertices are assigned weights according to the similarity between data nodes. Finally, an undirected weighted graph $G(V, E)$ is formed. Compared with traditional clustering algorithms such as K-means, the spectral clustering algorithm can solve the local optimal problem of convex sample space and can cluster in sample space of any shape, which has a better clustering effect. The specific process of spectral clustering can be summarized as follows.

1) The user behavior dataset should be cleaned and filtered first, then the similarity matrix $S \in R^{n \times n}$ is constructed by calculating the similarity of user behaviors.

2) The degree matrix D is created based on the similarity matrix S: The sum of the elements in each row of the matrix S is assigned to the element in the matrix D. Then, the Laplace matrix L is constructed as $L = D^{-1/2} S D^{-1/2}$.

3) The matrix L is decomposed into feature vectors, and appropriate feature vectors are selected for column storage to form a feature matrix Y.

4) Each vector in the feature matrix Y is taken as an independent sample, and the vectors are clustered using the K-means to form clusters like $C_1, C_2, ..., C_k$.

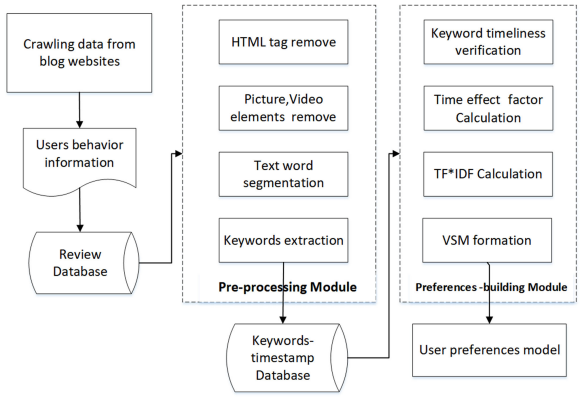

Fig. 1. User preference modeling process.

C. Similarity Calculation

In terms of user-based CF, the similarity evaluation of users is based on the rating matrix, and the *Pearson correlation coefficient* is used to measure the similarity. However, the rating matrix is usually sparse, especially, the ratings of different users for the same items are relatively sparse. As a result, the interests of users are significantly different, but the calculated similarity may be relatively close. Therefore, when using the Pearson correlation coefficient to calculate user similarity, the number of common interests of users should be considered, so a modified Pearson correlation coefficient is defined as

$$r\rho_{u,v} = \frac{|N(X \cap Y)|}{|N(X \cup Y)|} * \frac{\mathrm{cov}(X,Y)}{\sigma_X * \sigma_Y}, \tag{4}$$

where $r\rho_{u,v}$ denotes the similarity of users u and v, $\mathrm{cov}(X,Y)$ is the covariance of u and v for item ratings. σ_X and σ_Y denote the standard deviations of users u and v for ratings, respectively. Here, X and Y denote the set of items rated by u and v, respectively. Thus, $X \cap Y$ denotes the items rated by both u and v. $N(X \cup Y)$ denotes the number of items rated by users u or v. Also, $\frac{|N(X \cap Y)|}{|N(X \cup Y)|}$ is the proportion of common rating items of users u and v, which is used to modify the calculated Pearson coefficient.

IV. Proposed Scheme

A. Time-Aware Preference Model

The establishment of the user preference model is a key step for our recommendation scheme. The basic idea is to analyze the users' behavior recordings of online websites or applications. Based on probability statistics theory, the higher the frequency of terms, the higher the users' interest in them, and the user preference model can be established based on this theory. Fig. 1 shows the basic framework of building user preferences model which consists of three major steps.

Step 1: Crawling users' behavior recordings from Internet applications. In this step, a crawler program is developed to crawl behavior recordings, including users browsing records, comment records, post records, etc. Preprocessing users' behavior recordings from the database. In fact, users' recordings should be first filtered and cleaned to eliminate invalid recordings. Then HTML tags, picture elements, and video elements should be removed from documents.

Step 2: In this article, we use an attention-based method to build the interest model of users, so that the builded interest model can be dynamically adjusted as the recommended scenario changes. Assuming that $\mathrm{Content}_{(x_j)} = \{(kd_i, w_{1j}), (kd_2, w_{2j}), \ldots, (kd_n, w_{nj})\}$ is the target recommendation vector in TF-IDF form. The interest model is reversely activated based on the target vector $\mathrm{Content}_{(x_j)}$. Let

$$V_u^t = \begin{Bmatrix} \langle (kd_1^1, w_1^1), (kd_2^1, w_2^1), (kd_3^1, w_3^1), \ldots, (kd_n^1, w_n^1) \rangle_{\mathrm{seq}_1}^T \\ \langle (kd_1^2, w_1^2), (kd_2^2, w_2^2), (kd_3^2, w_3^2), \ldots, (kd_n^2, w_n^2) \rangle_{\mathrm{seq}_2}^T \\ \ldots \\ \langle (kd_1^n, w_1^n), (kd_2^n, w_2^n), (kd_3^n, w_3^n), \ldots, (kd_n^n, w_n^n) \rangle_{\mathrm{seq}_n}^T \end{Bmatrix}$$

is the behavior record matrix accumulated by user u at time t after TF-IDF operations, which can be used to build interest model of u. Here, we should calculate the similarity between the $\mathrm{Content}_{(x_j)}$ and each sequence seq_i in V_u^t, and we can use the vector similarity $\mathrm{Sim}_{(\mathrm{Content}_{x_j}, \mathrm{seq}_i)}$ to filter recordings. That is to say, if $\mathrm{Sim}_{(\mathrm{Content}_{x_j}, \mathrm{seq}_i)}$ smaller than a given threshold ∂, then seq_i will not be used in the modeling process. Moreover, we should use the calculated $\mathrm{Sim}_{(\mathrm{Content}_{x_j}, \mathrm{seq}_i)}$ as the weight to revise the TF-IDF weight corresponding to seq_i as

$\mathrm{seq}_i.\{\vec{w}\} = \mathrm{Sim}_{(\mathrm{Content}_{x_j}, \mathrm{seq}_i)} * \mathrm{seq}_i.\{\vec{w}\}$.

Finally, we will obtain a behavior recordings matrix S_u based on the $\mathrm{Content}_{(x_j)}$, and the definition is shown as

$$S_u = \begin{bmatrix} < (kd_1, w_1), \mathrm{timestamp}_1 >_u \\ < (kd_2, w_2), \mathrm{timestamp}_2 >_u \\ \ldots \\ < (kd_n, w_n), \mathrm{timestamp}_n >_u \end{bmatrix} \tag{5}$$

where S_u denotes a formatted behavior recordings of user u, and $\mathrm{timestamp}_k$ is the generation time of kd_k.

Step 3: Elements in S_u are sorted by timestamp, and the weight of keyword in S_u can be calculated according to (3). Then, the time impact factor of keyword could be defined as follows:

Definition 2 (Time impact factor): Let K_u be the keywords of user u. $K_u = \{kd_1, kd_2, kd_3, ..., kd_n\}$, where the timestamp of kd_i is h_i, and the current time is h_{now}, then the time factor of kd_i can be defined as

$$\mathrm{tif}_i = \frac{\ln\left(|h_i - h_{\mathrm{now}}|^{-1}\right)}{\sum_{kd_j \in K_u} \ln\left(|h_j - h_{\mathrm{now}}|^{-1}\right) + 1} \tag{6}$$

As in (6), tif_i is inversely proportional to the difference between h_i and h_{now}, that is to say, the closer the generation time of kd_i is to the current time h_{now}, the higher the tif_i should be. Thus, the weight of keyword should be redefined as $\beta_i = (\mathrm{tif}_i * \mathrm{TF\text{-}IDF}_{(t_i)})$, which can be described in detail as

$$\beta_i = \frac{\ln\left(|h_i - h_{\mathrm{now}}|^{-1}\right)}{\sum_{kd_j \in K_u} \ln\left(|h_j - h_{\mathrm{now}}|^{-1}\right)} * \left(\frac{n_{i,j}}{\sum_k n_{k,j}} * \log \frac{|D|}{|\{j : t_i \in d_j\}| + 1}\right). \tag{7}$$

Therefore, the user preference model can be defined as $PM = \{(kd_1, \beta_1), (kd_2, \beta_2), \ldots, (kd_n, \beta_n)\}$.

B. Feedback Mechanism

Recommendation models are generally based on the idea of when a user is interested in several items, he or she will remain absorbed in it for a long time. Subsequently, items of a similar type to users' preference will be recommended by systems. However, there are two disadvantages of this mode: 1) Ratings are often sparse, and it is hard to obtain full ratings of users for privacy concerns. 2) The offline training mode is difficult to ensure the real-time performance of the recommender system. To address these issues, we introduce a user feedback mechanism into our proposed scheme. As we know, the recommended items with a high click rate or browsing rate can be considered to be appropriate to the user's preferences. Thus the weight of such content should be increased. Meanwhile, the recommended items with low click or browsing rate may be considered to have a low matching degree with the user's preference model, and the weight of such content should be reduced. Specifically, the feedback mechanism can be divided into two phases: Building user feedback libraries and applying the feedback libraries. In terms of building user feedback libraries, the key steps are as follows.

Step 1. Identifying the Features of Feedback Data: After items have been recommended to users, it is necessary to track the feedback from users on the recommended items to optimize the model. The first thing is that we should identify the features of feedback data and store user feedback data into a database. Generally, we mainly consider the basic features of feedback data such as click or browsing rate, item tags, click, or browsing time in this work. As a result, a set of features $F = \{f_1, f_2, f_3, ..., f_n\}$ will be generated in this step.

Step 2. Classifying Feedback Data: After obtaining standardized feedback data, the next step is to classify these data according to recommendation effect. In this article, we define Rindex to indicate the user's interest in recommendations based on the feedback data, which can be calculated as

$$\text{Rindex} = \sum_{f_i \in F}^{n} w_{(f_i)} * S_{(f_i)} \tag{8}$$

where $w_{(f_i)}$ denotes the weight of feature f_i. In fact, based on the feedback data, the *information gain* of each $f_i \in F$ is calculated to denote the weight of f_i. Besides, we should digitize and normalize all of features (e.g., click or browsing rate, browsing duration, etc.) from the feedback data, $S_{(f_i)}$ is used to represent the score value of f_i after digitization and normalization. Here, we use the information gain of the feature f_i as the weight in (8), due to the *information gain* can express the contribution of f_i to the classification of records (e.g., features such as *click rate* and *browse duration* have different contributions on determining whether a record is negative or not). As for the used score $S_{(f_i)}$ in (8), it is essentially a normalized value of the specific feature (e.g., click rate is 0.43). The reason why we adopt the normalization method is that the value range and measurement of different features are different. Therefore, we need to map the values of different features to the same numerical space (i.e., [0,1]).

Algorithm 1: Building Feedback Libraries.

Input: DataSet denotes the user feedback to be processed; F denotes the defined set of features; λ is the threshold of positive library; γ is the threshold of negative library.
Output: boolean.

for d_i in DataSet **do**
  Δ Digitization and Normalization
  $(d_i) \rightarrow dn_i = \{(f_1, S_1), (f_2, S_2), \ldots, (f_n, S_n)\}$;
  for delement$_j$ **in** dn_i **do**
    if delement$_j$.$\{f\}$ in F **then**
      Rindex$_i$ += $w_{(f_j)}$*delement$_j$.$\{S\}$;
    end if
  end for
  if (Rindex$_i \geq \lambda$) **then**
    $dn_i \overset{\text{send}}{\rightarrow}$ positiveLibrary;
  end if
  if (Rindex$_i \leq \gamma$) **then**
    $dn_i \overset{\text{send}}{\rightarrow}$ negativeLibrary;
  end if
end for
return true;

After that, we can categorize the feedback data into the *positive* sample library and the *negative* sample library based on the feedback impact factor Rindex. Specifically, the positive sample library contains the items with positive feedback from users (i.e., users are more interested in these items), while, the negative sample library represents the items that are of little interest to users. The detail process of building feedback libraries is shown in Algorithm 1.

After completing the establishment of feedback libraries, then, we can describe the feedback mechanism in detail as follows.

1) When top-N recommendation items $RI = \{r_1, r_2, ..., r_n\}$ are generated by the recommender system, then α_i, which is the average similarity between $r_i \in RI$ and the *positive* library should be calculated. Based on the same principle, β_i, which is the average similarity between $r_i \in RI$ and the *negative* library should be calculated.

2) We set η as the threshold of positive similarity and set μ as the threshold of negative similarity. These threshold parameters are optimized by machine learning algorithms, but the specific process is not the focus of this article.

3) If $\alpha_i \geq \eta$, then r_i should be increased the recommendation weight by $(\alpha_i - \eta)$. Meanwhile, if $\beta_i \geq \mu$, then r_i should be reduced the weight by $(\beta_i - \mu)$.

Finally, the RI will be reconstructed based on the feedback mechanism. Specifically, the items in the RI will be reordered according to the adjusted recommendation weight, and the items with lower recommendation weight will be deleted from the top-N recommendation list.

C. Collaborative Filtering Based on Spectral Clustering

In this article, the CF approach based on spectral clustering can be divided into two stages: The user information clustering and the recommendation of items. The specific process can be described as follows.

The Stage of User Clustering

1) Constructing rating matrix M from users' rating data and the matrix M_{norm} is obtained by numerical normalization and smoothing of the rating matrix M.

2) For the matrix M_{norm}, the similarity of users is calculated as (4), then the similarity matrix of users R_{sim} is obtained.

3) The similarity matrix R_{sim} is input as a parameter of the spectral clustering algorithm, then the clustering result of users $C(c_1, c_2, ..., c_n)$ will be obtained.

It is important to note that the CF algorithm is also insensitive to time, therefore, in the process of using rating data to build a rating matrix M, we use the defined *time impact factor* in (6) to improve the timeliness of M. That is to say, for each rating element in M, we calculate its time impact factor and weight the rating value.

The Stage of Items Recommendation

1) Select the K-nearest neighbor set $U(u_1, u_2, ..., u_k)$ from the cluster containing the user u_t.

2) Calculate the prediction rating value $p_{u_t,\hat{\theta}}$ of user u_t for an unrated item $\hat{\theta}$ as

$$p_{u_t,\hat{\theta}} = \left(\sum_{u_i \in U}^{k} r\rho_{t,i} * p_{u_i,\hat{\theta}} \right) \Big/ k \tag{9}$$

where $r\rho_{t,i}$ denotes the similarity between user u_t and user $u_i \in U$, which can be calculated as (4). Then, $p_{u_i,\hat{\theta}}$ is the rating of user u_i for the item $\hat{\theta}$.

Finally, we can obtain the prediction rating vector of user u for all items as $P_{(CF_u)} = \{p_{u,1}, p_{u,2}, ..., p_{u,n}\}$. Here, $P_{(CF_u)}$ should be sorted and the top-N items in it can be selected as the recommendation items.

D. Content Filtering Based on User Preferences

Based on the constructed preference model PM, we use the content-based model to generate a batch of candidate recommendation items, which can effectively recommend new items that are not rated by other users. Since each user's preference model is based on his/her behavior recordings, then this method can eliminate the interference of other users' malicious behaviors (e.g., using multiple accounts to forge the ratings of an item) from interfering with the recommendation results. The basic idea of content filtering is to recommend items according to the user preference model. In this article, we use the *content filtering* method as one of the important components of our scheme. We can briefly introduce the process as follows.

1) Let kd_i be the ith keyword of item x_j. w_{ij} is the weight of kd_i on x_j, then the content of x_j can be defined as $\text{Content}_{(x_j)} = \{(kd_1, w_{1j}), (kd_2, w_{2j}), \ldots, (kd_n, w_{nj})\}$.

2) As mentioned above, the content filtering method recommends users with similar content to their previous favorite items. Therefore, it is necessary to model users' preferences based on their previous behaviors. In fact, we utilize the proposed approach in Section IV-B to build preference model as $PM_{(u)} = \{(kd_1, \beta_1), (kd_2, \beta_2), \ldots, (kd_n, \beta_n)\}$.

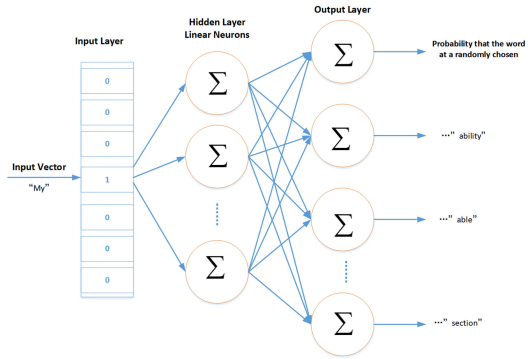

Fig. 2. Skip-gram model.

3) In general, we should map $PM_{(u)}$ and $\text{Content}_{(x_j)}$ to the same vector space, so that the recommendation calculation could be transformed into vector similarity calculation. One important thing is to calculate the semantic similarity of keywords between vectors. For instance, "algorithm" and "machine learning" have a high semantic correlation, but if we only use cosine similarity, the semantic similarity may be ignored. In this article, we use the Skip-gram model [33] to obtain word embeddings, and we use Google news corpus as the training dataset. The basic structure of Skip-gram is shown in Fig. 2, which is a three-layer neural network. The first layer is the input layer for keywords, and the input keyword is usually in vector form, we use the *one-hot encoding* technique to convert natural language keywords into vectors. The third layer is the output layer, which is the probability of other words appearing in the context when the input is known, and *Softmax* is usually used to calculate the probability in this layer. Besides, the hidden layer does not have any activate function and is usually composed of linear cells. We use the method of random gradient descent [33] to update the model parameters, and the generated word embedding is used to measure the semantic similarity of the input keyword. In fact, keywords are semantically similar, then their contexts are similar, and their vector representations are also similar.

4) After completing the attribute mapping, $PM_{(u)}$ and $\text{Content}_{(x_j)}$ will be unified into the same VSM. Then, we should calculate the similarity between the content vector $\text{Content}_{(x_j)}$ and the user preference vector $PM_{(u)}$ as

$$\rho_{u,x_j} = \frac{\sum_{i=1}^{n} (w_{ij} * \beta_i)}{\sqrt{\sum_{i=1}^{n} w_{ij}^2} * \sqrt{\sum_{i=1}^{n} \beta_i^2}} \tag{10}$$

where the cosine similarity is used to measure the degree of compliance with user preferences, and w_{ij} is the weight of the ith attribute in the vector $\text{Content}_{(x_j)}$. Finally, we can obtain the similarity vector of items as $P_{(CB_u)} = \{\rho_{u,x_1}, \rho_{u,x_2}, \ldots, \rho_{u,x_n}\}$. Here, $P_{(CB_u)}$ should be sorted and the top-N items in it can be selected as the recommendation items.

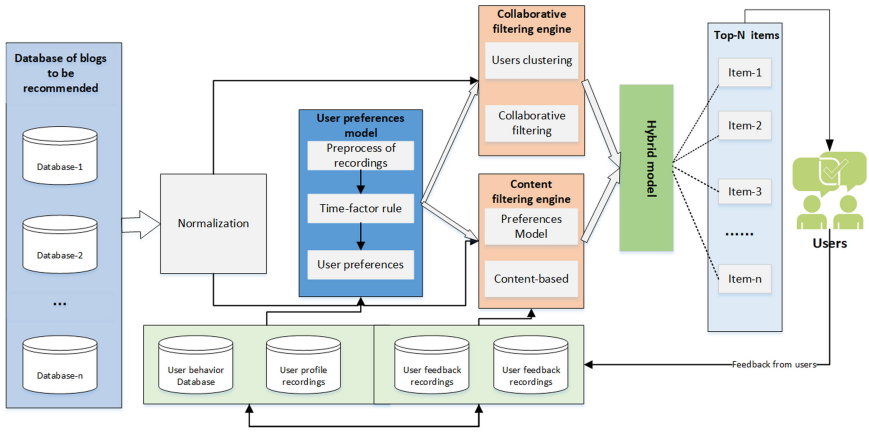

Fig. 3. Hybrid model.

E. Hybrid Model

In this section, a hybrid recommendation model based on logistic regression is proposed, and the architecture is shown in Fig. 3.

Definition 3 (Hybrid model): Let $D_r = \{d_1, d_2, ..., d_n\}$ be the recommendation item vector, and the vector of corresponding prediction rating based on CF for the user u as $P_{(CF_u)} = \{p_{u,1}, p_{u,2}, ..., p_{u,n}\}$. Meanwhile, the rating value of the content-based model is $P_{(CB_u)} = \{\rho_{u,1}, \rho_{u,2}, \ldots, \rho_{u,n}\}$. Then, let $FM = \{\omega_1, \omega_2, ..., \omega_n\}$ be the result of the hybrid model, where ω_i represents the final rating of recommended item d_i, which is calculated using $p_{u,i}$ and $\rho_{u,i}$.

Logistic regression is used to aggregate the results of recommendation both CF and content-based. Then $P_{(CF_u)}$ and $P_{(CB_u)}$ should be as the given parameters

$$\text{Sigmoid}(h_\theta(X)) = \frac{1}{1 + e^{-h_\theta(X)}} \tag{11}$$

where $h_\theta(X)$ can be defined

$$h_\theta(X) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n \tag{12}$$

where $(\theta_0, \theta_1, \ldots, \theta_n)$ denotes the constant coefficient for the input parameters $P_{(CF_u)} = \{p_{u,1}, p_{u,2}, ..., p_{u,n}\}$ and $P_{(CB_u)} = \{\rho_{u,1}, \rho_{u,2}, \ldots, \rho_{u,n}\}$, and $(\theta_0, \theta_1, \ldots, \theta_n)$ can be calculated by the gradient descent algorithm.

Overall, we should use the proposed hybrid approach to obtain a better recommendation effect and the specific process can be described as the following steps.

Step 1: Using the above CF in Section IV-C to obtain the recommendation items $D_r = \{d_1, d_2, \ldots, d_n\}$ and the corresponding prediction rating value $P_{(CF_u)} = \{p_{u,1}, p_{u,2}, \ldots, p_{u,n}\}$.

Step 2: Using the above content-based model defined in Section IV-D to obtain the recommendation items prediction rating value of D_r as $P_{(CB_u)} = \{\rho_{u,1}, \rho_{u,2}, \ldots, \rho_{u,n}\}$.

Step 3: $P_{(CF_u)} = \{p_{u,1}, p_{u,2}, \ldots, p_{u,n}\}$ and $P_{(CB_u)} = \{\rho_{u,1}, \rho_{u,2}, \ldots, \rho_{u,n}\}$ are selected as input parameters of (12). Then the comprehensive rating $FM = \{\omega_1, \omega_2, ..., \omega_n\}$ would be generated.

Step 4: $FM = \{\omega_1, \omega_2, ..., \omega_n\}$ will be sorted and the corresponding items $\partial_f = \{\tau_1, \tau_2, \ldots, \tau_n\}$ should be selected as the top-N recommendation items.

It is important to note that the obtained ∂_f should be filtered by feedback libraries (i.e., the positive and negative libraries in Section IV-B) to achieve better recommendations.

V. Experiments and Results

A. Data Preparation

To compare the proposed model with state-of-the-art methods, we use three public real-world datasets for comparison, which are introduced as follows.

1) *OULAD* (Open University Learning Analytics) dataset [31] is a recently released open-source dataset. The employed dataset (OULAD) contains 32 593 learners and their assessment results (about 10 655 280 records). In this article, we mainly focus on virtual learning environment (VLE) data, which show learner preference in choosing learning materials.
2) *MovieLens-Latest* This dataset is collected as part of the GroupLens Research Project of the University of Minnesota. This dataset consists of 27 000 000 ratings and 1 100 000 tag applications applied to 58 000 movies by 280 000 users.
3) *Book-Crossing* This dataset is collected by Cai-Nicolas Ziegler from the Book-Crossing community. The dataset contains 278 858 users (anonymized but with demographic information) providing 1 149 780 ratings (explicit/implicit) about 271 379 books.

In fact, we should take further measures to deal with the above datasets. Specifically, the data should be filtered to achieve more reliable and completed records. For instance, we obtain about 140 406 valid records from studentAssessment.csv by extracting the score is more than 60% from 173 913 records. Moreover, to reduce the interference of outlier data to the experiment, we use K-means to cluster the data and eliminate outlier data. A large abnormal number may cause great bias to the clustering result. For the Book-Crossing dataset, we remove about 8576 records from its Bx-Book-Ratings.csv. Then, the statistics of these three datasets are shown in Table I.

Furthermore, we should determine which basic features to be used based on the specific dataset. To be specific, we choose the online learning duration, the click rate of courses, and the

TABLE I
STATISTICS OF DATASETS

Dataset	OULAD	MovieLens-Latest	Book-Crossing
User number	32,094	3,681	78,858
Item number	848,575	9,743	271,380
Rating number	985,623	100,563	648,576
Rating density	0.076%	0.016%	0.012%

TABLE II
MEANINGS OF TP, FP, FN, AND TN

Items	Meanings
TP	Recommended by the system and approved by the user
FP	Recommended by the system but not approved by the user
FN	Not recommended by the system but approved by the user
TN	Not recommended by the system and not approved by the user

category of courses as the basic features in the OULAD dataset. As for the MovieLens-Latest dataset, we choose the browsing time, types of movies, and rating scores as the basic features. When it comes to the Book-Crossing dataset, the rating score from users, the publication date, and the item tag are chosen as the basic features.

B. Evaluation Metrics

In this article, precision, recall, and F1-score are utilized as metrics to evaluate our proposed scheme. The definitions are as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}},\ \text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$$

$$\text{F1-score} = 2 \times \frac{\text{recall} \times \text{precision}}{\text{recall} + \text{precision}}$$

where *Precision* is the ratio of the correctly judged existent propagation relations to all the judged existent propagation relations. *Recall* is the ratio of the correctly judged existent propagation relations to all the existent propagation relations in the system. *F1-score* is the harmonic mean (average) of the precision and recall. Hence, F1-score will be a better measure when precision and recall are sometimes contradictory. The meanings of TP, FP, and FN are described in Table II.

C. Evaluation Baselines

In this section, we compare the proposed scheme with several state-of-the-art recommendation algorithms. Thus, the used base-lines in our experiments as follows.

1) *PRMR [34]:* This method is proposed to improve the efficiency of CF for movie recommendations, then a simple but high-efficient recommendation algorithm is proposed, which exploits users' profile attributes to partition them into several clusters. For each cluster, a virtual opinion leader is conceived to represent the whole cluster so that the dimension of the original user-item matrix can be significantly reduced.
2) *AROLS [35]:* This article introduces a learning style model to represent features of online learners. It also presents an enhanced recommendation method named adaptive recommendation based on online learning style (AROLS), which implements learning resource adaptation by mining learners' behavioral data. AROLS applies CF and association rule mining to extract the preferences and behavioral patterns of each cluster.
3) *HRBRM [36]:* The most important achievement of this article is to present a novel approach in hybrid recommendation systems, which identifies the user similarity neighborhoods from implicit information.
4) *HRSRL [37]:* This article proposed a hybrid recommendation system, combining content-based and CF for job recommendations. In this proposed system, statistical relational learning (SRL) is used to combine the two recommendation approaches through its ability to directly represent the probabilistic dependencies among the attributes of related objects.

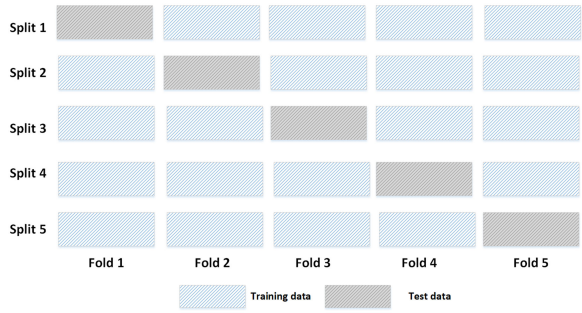

Fig. 4. Process of the adopted cross-validation.

Note that for all comparison methods, we tune the hyperparameters carefully according to corresponding references to ensure that each method achieves its performance for a fair comparison. To be specific, we divide the employed dataset into five parts, which can be test datasets and training datasets. Moreover, we set the test dataset accounts for $1/5$ (20%), and the training data takes up $4/5$ (80%) of the original dataset. Five rounds of training and testing are required, and each round of the training process needs to change the test dataset (keep the number of test dataset unchanged, accounting for $1/5$ (20%) of the total dataset), so that after five rounds of testing, the total employed dataset can be utilized. Here, we can briefly describe this process, as shown in Fig. 4.

D. Impact of Parameters

To make our scheme achieve better performance, we analyze how the parameters affect the performance of our proposed scheme on all the employed datasets (i.e., OULAD, MovieLens-Latest, and Book-Crossing). We study the hyperparameters of λ and γ, which are, respectively, the threshold of building positive and negative libraries. We sample the values of λ and γ all from 0.1 to 0.9, and the results on F1-score are shown in Fig. 5.

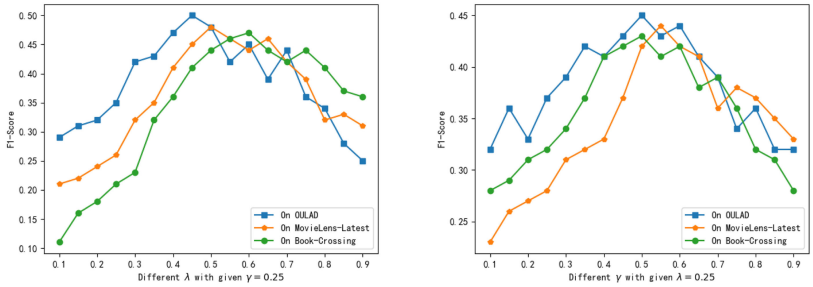

Fig. 5. Parameters study on γ and λ. (a) Parameter study on λ. (b) Parameter study on γ.

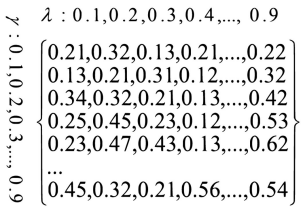

Fig. 6. Selection process of λ and γ.

TABLE III
RESULTS OF TRAINING PROCESS ON DIFFERENT DATASETS

Dataset	λ	γ	F1-score
OULAD	0.6	0.5	0.63
MovieLens-Latest	0.7	0.6	0.53
Book-Crossing	0.6	0.7	0.56

Specifically, we first evaluate the impact of parameter λ on the recommendations. We set $\gamma = 0.25$ and Top-N=25 (i.e., the number of recommendation items). F1-score increases when λ increases, which conforms to the fact that our scheme utilizes more information from user preferences. However, when λ increases to a certain value, the corresponding F1-score decreases, this may be the accuracy of the feedback library decreases. Based on the same principle, we also set $\lambda = 0.25$ and Top-N=25 to evaluate the impact of γ, and the results are shown in Fig. 5 (b). Furthermore, to select the appropriate λ and γ, we conduct an offline training process of parameters, which maximizes the F1-score of recommendation results by constantly changing λ and γ.

As illustrated in Fig. 6, we keep adjusting λ and γ constantly to achieve better recommendations (F1-score) and finally obtain the optimized parameters in Table III through several rounds of offline training.

E. Experimental Results

In our experiments, the selection of internal parameters (i.e., λ and γ) are based on the results in Table III. For performance comparison, we report the recommendation precision and recall of different methods over the three datasets in Table IV. It is important to note that all the results in Table IV are the mean performance of each experiment five times.

From the results in Table IV, we have the following insightful observations: First, our proposed scheme performs better than other baselines evaluated here on the three employed datasets in terms of precision and recall. To be specific, we can see that our scheme performs better than PRMR about 11.6% and 9.2% on metrics of precision and recall, respectively. In fact, PRMR is used to improve the efficiency of CF in movie recommendation scenes, the main focus is to improve the time complexity of CF but the real-time performance of recommendations and user preferences are not fully considered. As for the AROLS model, which performs well in the dataset of OULAD, since this method focuses on the recommendation of learning resources, but it just applies CF and association rule mining to extract the preferences without using any user feedback, which leads the precision and recall are still lower than that of our scheme about 15.8% and 8.5%, respectively. Second, from the results, we find that hybrid methods perform better than traditional recommendation algorithms. For instance, HRBRM performs better than PRMR and AROLS by about 4.5% and 4.9% in terms of recall. HRSRL also performs better than PRMR and AROLS by about 4.8% and 10.1% on the metric of precision, respectively. This is because these hybrid schemes take advantage of CF and content-based approaches. Compared with the above-mentioned hybrid models, our scheme achieves better performance on these three datasets. Specifically, for all the used datasets, our scheme performs better than HRBRM and HRSRL by about 7.2% and 6.6% on the metrics of F1-score, respectively. Note that the F1-score that we used is the average value of different Top-N conditions in Table IV. Especially, with the increase of recommendations, the performance stability of our scheme is superior to the other two baselines. In fact, with the increasing times of experiments, the advantages of our scheme in time and user preference perception will be more obvious. The reasons may be that our proposed scheme with the time-aware and user feedback mechanisms is more sensitive to user preferences.

F1-score measures the overall performance of recommendation schemes, and it is shown in Fig. 7. For our proposed scheme, F1-score changes slightly with the number of recommendation items, while the baselines also change around certain values. To be specific, we can measure the stability of our scheme by calculating the standard deviation of F1-score. For the proposed scheme, the calculated standard deviation of F1-score is about 0.0401, which is in a low state. We can see that the performance of our scheme is in a relatively stable state compared with other used hybrid schemes (i.e., the standard deviation of HRBRM

TABLE IV
EVALUATIONS OF PRECISION AND RECALL

Top-N	DataSet	Our scheme		PRMR		AROLS		HRBRM		HRSRL	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
10	OULAD	**0.732**	**0.572**	0.712	0.569	0.722	0.495	0.682	0.562	0.717	0.549
	MovieLens-Latest	0.601	0.551	**0.709**	0.563	0.577	0.574	0.673	0.567	0.651	**0.621**
	Book-Crossing	**0.632**	**0.569**	0.554	0.446	0.585	0.477	0.579	0.526	0.619	0.451
20	OULAD	**0.733**	0.452	0.611	0.491	0.679	0.493	0.673	**0.562**	0.605	0.432
	MovieLens-Latest	0.493	**0.556**	0.467	0.527	0.524	0.468	0.546	0.492	**0.569**	0.453
	Book-Crossing	**0.682**	0.514	0.596	0.521	0.642	0.452	0.608	**0.533**	0.581	0.482
30	OULAD	**0.705**	0.534	0.583	0.461	0.568	0.453	0.694	0.521	0.661	**0.573**
	MovieLens-Latest	**0.631**	0.489	0.516	0.507	0.551	0.431	0.564	0.516	0.531	**0.578**
	Book-Crossing	**0.629**	**0.527**	0.588	0.451	0.572	0.483	0.619	0.491	0.596	0.448
40	OULAD	**0.691**	**0.564**	0.541	0.435	0.536	0.516	0.532	0.465	0.625	0.448
	MovieLens-Latest	0.624	0.533	0.581	0.523	0.551	**0.556**	0.646	0.503	**0.677**	0.538
	Book-Crossing	**0.596**	0.501	0.538	**0.525**	0.451	0.456	0.531	0.461	0.565	0.473
50	OULAD	**0.696**	**0.592**	0.551	0.431	0.501	0.482	0.561	0.497	0.603	0.475
	MovieLens-Latest	**0.623**	**0.515**	0.528	0.498	0.492	0.495	0.526	0.432	0.562	0.481
	Book-Crossing	**0.617**	**0.566**	0.511	0.449	0.513	0.531	0.558	0.504	0.558	0.476
60	OULAD	**0.579**	0.495	0.568	0.451	0.493	0.485	0.529	**0.562**	0.535	0.501
	MovieLens-Latest	0.595	**0.575**	0.609	0.542	0.505	0.507	0.482	0.547	**0.612**	0.462
	Book-Crossing	**0.588**	0.453	0.557	0.489	0.489	0.453	0.476	**0.526**	0.501	0.469

Bold entities in Table IV indicate the maximum precision and recall obtained in a given test.

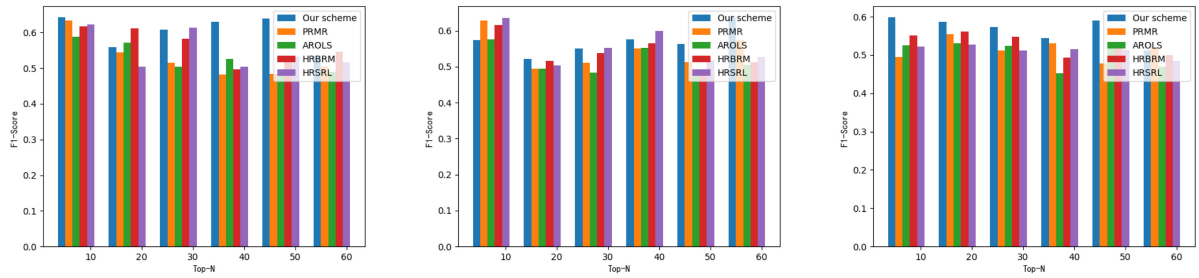

Fig. 7. Results of F1-score on all used datasets. (a) F1-score on OULAD. (b) F1-score on MovieLens-Latest. (c) F1-score on Book-Crossing.

is 0.0428 and that of HRSRL is 0.0459). As shown in Fig. 7, our scheme achieves the best performance for F1-score on all the three datasets. Specifically, the average F1-score of our scheme is about 10.2%, 12.5%, 7.2%, and 6.6% greater than that of PRMR, AROLS, HRBRM, and HRSRL, respectively. Overall, compared with PRMR and AROLS, our scheme can make use of the advantages of CF and content-based. For the hybrid baselines (i.e., HRBRM, HRSRL), our scheme takes into account the time characteristics of user preferences and the user feedback on recommendation items. Furthermore, to verify that the improvements in our scheme are statistically significant, we conduct several t-tests to analyze the achieved improvements. To be specific, we adopt the paired-sample t-tests and use the calculated F1-score of baselines as the analysis data. Thus, we can briefly describe the process of t-test as follows. First, we should propose hypothesis and test level as: H_0: $\mu_d = 0$ (null hypothesis); H_1: $\mu_d \neq 0$ (alternative hypothesis); Two-sided test, test level: $\alpha = 0.05$. Then, we can calculate the test statistics as: $t = \frac{\bar{d}-\mu_d}{\frac{S_d}{\sqrt{n}}}$, $S_d = \sqrt{\frac{\sum d^2 - (\sum d)^2/n}{n-1}}$. Thus, we obtain the test statistics as: $t_{\text{HRBRM}} = 2.97$, $t_{\text{HRSRL}} = 3.55$, $t_{\text{PRMR}} = 4.15$, and $t_{\text{AROLS}} = 6.41$. Note that the calculation process is complicated and not the focus of this article, so we give the results directly. Finally, we query the t-tests threshold table and find that $t_{0.05/2,17} = 2.11$. Due to all of the calculated t (i.e., t_{HRBRM}, t_{HRSRL}, t_{PRMR}, and t_{AROLS}) is bigger than $t_{0.05/2,17}$, then H_0 should not be approved, and all the improvements of our scheme are statistically significant.

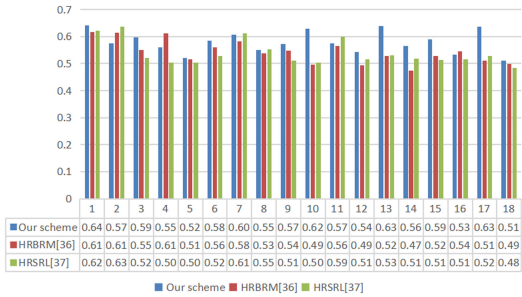

Fig. 8. Comparisions of F1-score.

1) Stability of Performance: We have adopted multiple Top-N tests to study whether the performance of our scheme is relatively stable. Hence, we analyze the F1-score of our scheme, considering its change trend and fluctuation with the recommendation size. Then, we have plotted the F1-score comparison histogram of all the involved hybrid schemes. As shown in Fig. 8, the average F1-score of our scheme is $\bar{u} = 0.579$, and we can calculate the standard deviation of F1-score for our scheme as $\delta = \sqrt{(x_1-\bar{u})^2 + (x_2-\bar{u})^2 + \ldots (x_n-\bar{u})^2} = 0.0401$. We see that the performance of our scheme is relatively stable with the recommendation size. Note that the standard deviation of

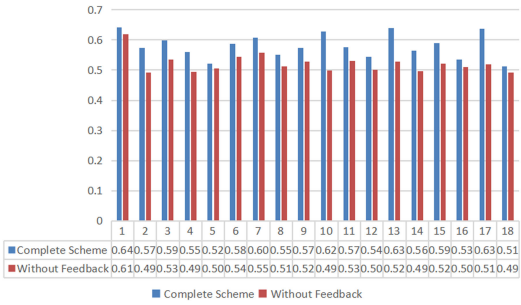

Fig. 9. Evaluations of the feedback mechanism.

Fig. 10. Evaluations of the time factor.

other hybrid schemes (HRBRM, HRSRL) is 0.0428 and 0.0459, respectively.

2) Effect of Our Feedback Mechanism: To verify the effect of the feedback mechanism in our proposed scheme, we have designed and implemented a set of experiments, which includes the proposed complete scheme and the proposed scheme without the feedback mechanism. In this test, we use F1-score as the metric of performance, and the experimental results are shown in Fig. 9.

From the results in Fig. 9, we can see that the performance of the scheme without the feedback mechanism is significantly lower than that of the complete scheme. As far as F1-score is concerned, the complete scheme obtains higher scores than that of the scheme without the feedback mechanism on all datasets. In addition, the average F1-score of the complete scheme is $\bar{F_c} = 5.79$, and the average value of the scheme without the feedback mechanism is $\bar{F_w} = 4.91$. Thus, the average F1-score of the complete scheme is about 17.9% higher than that of the scheme without the feedback mechanism. Therefore, whether from the test results on each dataset or the overall average performance, the feedback mechanism can significantly improve the recommendation results.

3) Effect of Time Factor: To evaluate the effect of the time factor in our scheme, we have designed and implemented the control experiment of the complete scheme and the scheme without the time factor mechanism. In this test, we use F1-score as the metric of performance, and the results are shown in Fig. 10. Thus, we can see that the complete scheme has better performance in F1-score than that of the scheme without the time factor mechanism. To be specific, the complete scheme obtains higher values than that of the scheme without the time factor on each dataset. Also, the average F1-score of the complete scheme is $\bar{F_c} = 0.579$, and the average value of the scheme without the time factor mechanism is $\bar{F_t} = 0.484$. In fact, the average F1-score of the complete scheme is about 19.2% higher than that of the scheme without the time factor mechanism. Therefore, the time impact factor can improve the performance of the proposed scheme obviously.

TABLE V
COMPARISON OF RESOURCES CONSUMPTION

Schemes	Top-N	MEM(MB)	CPU(%)
Our Scheme	20	0.52	1.4
	40	1.13	2.1
	60	1.32	4.4
HRBRM [36]	20	5.53	5.7
	40	6.96	12.4
	60	11.2	14.5
HRSRL [37]	20	2.79	5.3
	40	3.14	8.9
	60	6.47	10.2

4) Resources Consumption: Furthermore, to evaluate the resource consumption of our proposed scheme, we have designed and implemented a comparison experiment. This experiment is built on a desktop computer (Lenovo ThinkCentre M720) with Intel Core i7-7500 2.7 GHz processor and 32 GB. Specifically, we evaluate the resource consumption of algorithms in terms of memory consumption and CPU usage, then use VisualVM 1.4.4 as a measurement tool to monitor the memory and CPU usage.

As shown in Table V, we mainly consider the resources consumption of the involved hybrid models (i.e., HRBRM [36] and HRSRL [37]) and our proposed scheme. We can see that our scheme performs better than the other two models, the memory usage of our scheme is less than 2 MB, while for the HRBRM [36] and HRSRL [37] schemes, the minimum memory usage is 5.53 and 2.79 MB, respectively. In terms of the CPU usage, our proposed scheme is significantly lower than that of HRBRM [36] and HRSRL [37], respectively. To be specific, for our consensus, the highest CPU usage of our scheme is 4.4% (Top-N=60), which is lower than the lowest value of HRBRM [36] (i.e., 5.7%) and HRSRL [37] (i.e., 5.3%). This is because that we adopt the efficient and simple VSM model, which involves less inner product computations. Also, the hybrid approach in our scheme is based on the logistic regression, which does not involve complex computations compared with other comparisons.

5) Threats to Validity: Due to our scheme is based on content-based and CF, the effect of this scheme is limited for scenes that are not suitable for content recommendation, such as videos and pictures recommendations.

VI. CONCLUSION

In this article, we proposed a hybrid recommendation scheme combining content-based and CF. To improve the accuracy of the hybrid model, we propose the time impact factor of user preferences and analyze the effect of the time factor on recommendations. Also, a user feedback mechanism is proposed in this article, and such a mechanism is used to filter the final recommendations. Simulation results demonstrate that the proposed scheme is effective compared with baselines.

As future work, we will introduce the factorization machine (FM) method into our scheme to improve the recall and precision of the proposed model in this article. In addition, we will further study how to filter out invalid feedbacks according to different recommendation scenarios to improve the efficiency of the feedback mechanism.

REFERENCES

[1] "Tiktok recommendation algorithm analysis," 2019. [Online]. Available: https://zhuanlan.zhihu.com/p/88815475.html

[2] B. Peizhen, G. Yan, L. Fangling, and L. Haiping, "Joint interaction with context operation for collaborative filtering," *Pattern Recognit.*, vol. 88, pp. 729–738, 2019.

[3] G. Virane, S. Upasana, S. Mayank, and K. Sunil Kumar, "A noval collaborative filtering technique approach for recommender system," in *Proc. Amity Int. Conf. Artif. Intell.*, 2019, pp. 151–155.

[4] A. Lokhande and J. Pooja, "Hybrid collaborative filtering model using hierarchical clustering and PCA," in *Proc. Recent Adv. Int. Trends Eng. Appl.*, 2019, pp. 121–125.

[5] L. Feng and G. Wei-Wei, "Research on recommendation system algorithm based on deep learning mode in grid environment," in *Proc. Int. Conf. Robots Intell. Syst.*, 2019, pp. 250–253.

[6] W. Peng and X. Baogui, "SPMF: A social trust and preference segmentation-based matrix factorization recommendation algorithm," *CoRR*, 2019. [Online]. Available: https://arxiv.org/abs/1903.04489

[7] W. Dan, "Music personalized recommendation system based on hybrid filtration," in *proc. Int. Conf. Intell. Transp., Big Data Smart City*, 2019, pp. 430–433.

[8] J. Zhao *et al.*, "Attribute mapping and autoencoder neural network based matrix factorization initialization for recommendation systems," *Knowledge-Based Syst.*, vol. 166, pp. 132–139, 2019.

[9] P. Hashem, M. Parham, and E. Shahrokh, "TCFACO: Trust-aware collaborative filtering method based on ant colony optimization," *Expert Syst. Appl.*, vol. 118, pp. 152–168, 2019.

[10] S. Cheng and W. Wanyan, "Rating prediction algorithm based on user time-sensitivity," *Information*, vol. 11, no. 1, 2020, Art. no. 4.

[11] Li Yu, Lu Liu, and L. Xuefeng, "A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in e-commerce," *Expert Syst. Appl.*, vol. 28, no. 1, pp. 67–77, 2005.

[12] R. Logesh and V. Subramaniyaswamy, "Exploring hybrid recommender systems for personalized travel applications," in *Proc. Cognitive Inform. Soft Comput.*, 2019, pp. 535–544.

[13] M. Gandhi and G. Sonali, "An enhanced approach for tourism recommendation system using hybrid filtering and association rule mining," *Asian J. Convergence Technol.*, vol. 36, no. 1, pp. 1–4, 2019.

[14] L.-Y. Leong, T.-S. Hew, K.-B. Ooi, and C. A. Yee-Loong, "Predicting the antecedents of trust in social commerce–a hybrid structural equation modeling with neural network approach," *J. Bus. Res.*, vol. 110, pp. 24–40, 2020.

[15] X.-L. Zheng, C.-C. Chen, J.-L. Hung, H. Wu, F.-X. Hong, and L. Zhen, "A hybrid trust-based recommender system for online communities of practice," *IEEE Trans. Learn. Technol.*, vol. 8, no. 4, pp. 345–356, Jan. 2015.

[16] H. Lee and J. Um, "A study on the context-aware hybrid Bayesian recommender system on the mobile devices," *IAENG Int. J. Comput. Sci.*, vol. 45, no. 1, pp. 40–45, 2018.

[17] A Salehi-Abari and B. Craig, "Preference-oriented social networks: Group recommendation and inference," in *Proc. 9th ACM Conf. Recommender Syst.*, 2015, pp. 35–42.

[18] W. Xiaoyan, S. Lifeng, W. Zhi, and D. Meng, "Group recommendation using external followee for social tv," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2012, pp. 37–42.

[19] S. Lifeng, W. Xiaoyan, W. Zhi, Z. Hong, and W. Zhu, "Social-aware video recommendation for online social groups," *IEEE Trans. Multimedia*, vol. 19, no. 3, pp. 609–618, Mar. 2017.

[20] L. Huizhi, X. Yue, T. Dian, and P. Christen, "Time-aware topic recommendation based on micro-blogs," in *Proc. 21st ACM Int. Conf. Inf. knowl. Manage.*, 2012, pp. 1657–1661.

[21] G. Shipra, B. Muskan, and A. Sinha, "Modeling recommendation system for real time analysis of social media dynamics," in *Proc. IEEE 11th Int. Conf. Contemporary Comput.*, 2018, pp. 48–55.

[22] P. Enrico, R. Giuseppe, and R. Troncy, "Entity2rec: Learning user-item relatedness from knowledge graphs for top-N item recommendation," in *Proc. 11th ACM Conf. Recommender Syst.*, 2017, pp. 32–36.

[23] L. M. D. Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, "Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks," *Int. J. Approx. Reasoning*, vol. 51, no. 7, pp. 785–799, 2010.

[24] L. Chun, L. Philippe, and M. Stankovic, "Travel attractions recommendation with knowledge graphs," in *Eur. Knowl. Acquisition Workshop*, 2016, pp. 416–431.

[25] R. Catherine and W. Cohen, "Personalized recommendations using knowledge graphs: A probabilistic logic programming approach," in *Proc. 10th ACM Conf. Recommender Syst.*, 2016, pp. 325–332.

[26] H. Wang, N. Wang, and D. Y. Yeung, "Collaborative deep learning for recommender systems," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 1235–1244.

[27] K.-L. Skillen, C. Liming, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim, "Ontological user modelling and semantic rule-based reasoning for personalisation of help-on-demand services in pervasive environments," *Future Gener. Comput. Syst.*, vol. 34, pp. 97–109, 2014.

[28] Cataldo Musto, "Enhanced vector space models for content-based recommender systems," in *Proc. 4th ACM Conf. Recommender Syst.*, 2010, pp. 361–364.

[29] L. Shangsong, Z. Xiangliang, R. Zhaochun, and E. Kanoulas, "Dynamic embeddings for user profiling in twitter," in *Proc. 24th ACM SIGKDD Int. Conf. Know. Discovery Data Mining*, 2018, pp. 1764–1773.

[30] Y. Lu, L. Chuang, and Z. Zhang, "Multi-linear interactive matrix factorization," *Knowl. Based Syst.*, vol. 85, pp. 307–315, 2015.

[31] *Open university learning analytics dataset*, 2017. [Online]. Available: https://www.nature.com/articles/sdata2017171/

[32] Y. Lu, H. Junming, Z. Ge, L. Chuang, and Z. Zhang, "TIIREC: A tensor approach for tag-driven item recommendation with sparse user generated content," *Inf. Sci.*, vol. 411, pp. 122–135, 2017.

[33] L. Jarvan, H. H. Zhuo, H. Junhua, and E. Rong, "LTSG: Latent topical skip-gram for mutually learning topic model and vector representations," *CoRR*, 2017. [Online]. Available: https://arxiv.org/abs/1702.07117

[34] Z. Jiang, W. Yufeng, Y. Zhiyuan, and Q. Jin, "Personalized real-time movie recommendation system: Practical prototype and evaluation," *Tsinghua Sci. Technol.*, vol. 25, no. 2, pp. 180–191, 2020.

[35] C. Hui, Y. Chuantao, L. Rumei, R. Wenge, X. Zhang, and B. David, "Enhanced learning resource recommendation based on online learning style model," *Tsinghua Sci. Technol.*, vol. 25, no. 3, pp. 348–356, 2020.

[36] A. A. Kardan and M. Ebrahimi, "A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups," *Inf. Sci.*, vol. 219, pp. 93–110, 2013.

[37] Y. Shuo, K. Mohammed, A. Khalifeh, G. Trey, and S. Natarajan, "Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach," *Knowl. Based Syst.*, vol. 136, pp. 37–45, 2017.

Hongzhi Li received the M.S. degree from the School of Computer Science and Engineering, Wuhan University of Technology, Wuhan, China, in 2014. He is currently working toward the Ph.D. degree with Shanghai Maritime University, Shanghai, China.

His research interests include recommender systems and information security.

Dezhi Han (Member, IEEE) received the B.S. degree from Hefei University of Technology, Hefei, China, in 1990, and the M.S. and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2001 and 2005, respectively.

He is currently a Professor of computer science and engineering with Shanghai Maritime University, Shanghai, China.