A Time-Aware Hybrid Recommendation Scheme Combining Content-based and Collaborative Filtering

Hongzhi Li, 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 paper, 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 development of the Internet, users can enjoy rich information services and convenient social interaction through Internet applications, such as Twitter, Taobao, and TikTok [1], etc. At the same time, the problem of information overload of Internet applications is becoming more and more serious, which makes it difficult for users to choose what they really like. Therefore, various kinds of recommendation models are widely used to help users locate information. In general, recommendation models are utilized to recommend items for users by predicting their preferences, and specific categories of recommendation models could be divided into collaborative filtering, content-based, and hybrid approaches.

The collaborative filtering approach [2]-[4] is based on the view that the higher the similarity between users, the more the user preferences overlap. So it is reasonable to recommend items for similar users according to their preferences, The content-based approach [5], [6] is based on representations to recommend items. Item representations are usually exacted from item descriptions, and it is necessary to calculate the similarity between item representations and user profiles. Hybrid approach [9] generates recommendations by combining several other approaches, this approach is based on the idea that the hybrid method should take advantage of other approaches, and avoid disadvantages of each approach to achieve a better recommendation effect. This approach is usually based on collaborative filtering and content-based models, with the help of machine learning algorithms.

Among these approaches, collaborative filtering is used widely in the field of e-commerce for recommending food, books, and movies. Content-based performs well in recommending blogs, news, and documents. Generally, the collaborative filtering has a better performance than the content-based model. But the better performance is based on a sufficient number of user information, including personal information and behavior information. Moreover, the collaborative filtering model usually suffers from cold start problems 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, in the beginning, 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 with time.

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, which changes and adopts different recommendation technologies according to the background and actual situations or requirements of problems. According to our experience, the time factor has a significant impact on user preferences, and users' profiles are usually changing with time, thus the updating of user preferences should be paid more attention to. Moreover, we should improve the recommendation model according to the feedback from users.

In this paper, a novel hybrid recommendation model based on content-based and collaborative filtering is proposed, which contains 3 key points.

- For building user profiles, we use the time factor as a weighted basis for selecting behavior records. User behavior records with high real-time performance will be better utilized.
- We introduce a feedback mechanism into the model to improve the recommendation effect and then optimize the model according to the user feedback.
- Spectral clustering algorithm is used to improve the efficiency of collaborative filtering.

Finally, we use logistic regression to optimize the recommendation results.

Hongzhi Li and Dezhi Han (Corresponding Author) are with the College of Information Engineering at Shanghai Maritime University.

The rest of the paper is organized as follows. In Section 2 the related works are presented. Section 3 provides the background necessary to understand the proposed scheme, such as the preference representation and the spectral clustering algorithm. In Section 4, we describe the proposed scheme in detail, including the definition of time impact factor and our proposed feedback mechanism. In Section 5, we introduce the experimental environment and analyze the experimental results. Finally, the conclusions and future work are introduced in Section 6.

II. Related work

The main purpose of recommendation models is to provide helpful and suitable items for users. The traditional recommendation approaches include collaborative filtering, content-based, and knowledge-based, etc. Specifically, these approaches mainly focus on the fields of online news, social media, online advertising, and e-commerce, etc. However, a single recommendation approach may perform not well enough, and it is difficult to collect detailed 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 authors of [11] present a hybrid model of collaborative filtering based on items and users. This model combines both item-based and user-based collaborative filtering. The similarity calculation between active users and other neighbor users is based on other items related to prediction items, not on all items. Then the predicted item and items used to predict are similar in content. The research in [12]-[14] introduces a hybrid model, which employs user-based similarity, POI-based (Point-Of-Interest) similarity and geographic information to recommend tourist spots. In [15], the authors 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. In [16], [23], the authors proposed a bayesian network model combining content-based and collaborative filtering, 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 Micro-blogs. Since hot topics of micro-blogging 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. 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. With the popularity of artificial intelligence technology, more and more systems use this technology to improve the recommendation efficiency. In the work of [22], [24], knowledge graphs are mainly integrated into the recommendation generation process as a data set 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 multi-linear interactive MF (Matrix Factorization) 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 (Matrix Factorization) 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 focuses 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 work is built on the prior work of content-based and collaborative filtering, and we consider particularly the time factor and user feedback to enhance the performance of recommendation.

III. PRELIMINARIES

In order to better understand our proposed scheme, this section describes the necessary background and the main technologies on which the proposed scheme is based.

A. Preference Representation

User preference should be represented in a way that can be easily processed by the computer system. Usually, natural language descriptions of users should be converted to structures that computers can process directly. There are several structures used widely, such as user-item rating matrix [7], user-interests knowledge table [10], keywords vector, VSM (Vector Space Model) and semantic ontology [27]-[29]. In this work, we choose VSM as the user preference structure. Here, we first briefly introduce several concepts about VSM as follows.

- *Document*: It is usually a fragment with a certain scale in an article, such as sentence, sentence group, paragraph and paragraph group.

- *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. A document is regarded as a collection of features it contains, which is expressed as $Document = d(t_1, t_2, \ldots, t_n)$, where t_k is a feature term, and $1 \leq k \leq n$.
- *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 $d(t_1, w_1; t_2, w_2; \ldots; t_n, w_n)$.

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.

TF-IDF (term frequency-inverse document frequency) [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.

Step1: *TF* (term frequency) means term frequency, that is, the number of times a term appears in an article. It can be calculated as Eq. (1).

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

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.

Step2: *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 Eq. (2).

$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}| + 1}, \tag{2}$$

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.

Step3: We can calculate the TF-IDF value of the term t_i as $tf_{ij} * idf_i$, which is shown in Eq. (3).

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

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 k-th preference.

B. Spectral clustering

Generally, the similarity between users or items should be calculated in the collaborative filtering algorithm. For instance, in user-based collaborative filtering, 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:

- 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.
- 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 $L = D^{-1/2} S D^{-1/2}$.
- The matrix L is decomposed into feature vectors, and appropriate feature vectors are selected for column storage to form a feature matrix Y.
- 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.

C. Similarity Calculation

In terms of user-based collaborative filtering, 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 Eq. (4).

$$r\rho_{u,v} = \frac{|N(u) \cap N(v)|}{|N(u)| \cup |N(v)|} * \frac{\mathrm{cov}(u, v)}{\sigma_u * \sigma_v}, \tag{4}$$

where $r\rho_{u,v}$ denotes the similarity of users u and v, $cov(u, v)$ is the covariance of u and v for item ratings. σ_u and σ_v denote the standard deviations of users u and v for ratings, respectively. Then, $\frac{|N(u) \cap N(v)|}{|N(u)| \cup |N(v)|}$ is the proportion of common rating items of users u and v, which is used to modify the calculated Pearson coefficient, $N(u)$ and $N(v)$ denote the number of items that rated by users u and v respectively.

IV. The 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:

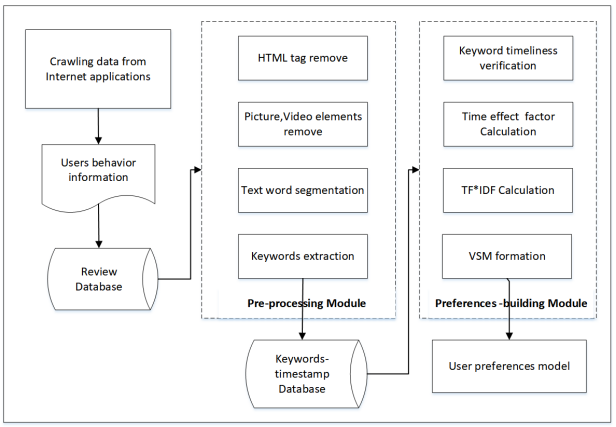


Fig. 1: User preference modeling process.

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 work, 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 $Content_{(x_j)} = \{(kd_i, w_{1j}), (kd_2, w_{2j}), \ldots, (kd_n, w_{nj})\}$ is the target recommendation vector in TF-IDF form. The interest model is reversely activated based on the target vector $Content_{(x_j)}$. Let

$$V_u^t = \left\{ \begin{array}{l} \langle (kd_1^1, w_1^1), (kd_2^1, w_2^1), (kd_3^1, w_3^1), \ldots, (kd_n^1, w_n^1) \rangle_{seq_1}^T \\ \langle (kd_1^2, w_1^2), (kd_2^2, w_2^2), (kd_3^2, w_3^2), \ldots, (kd_n^2, w_n^2) \rangle_{seq_2}^T \\ \cdots \\ \langle (kd_1^n, w_1^n), (kd_2^n, w_2^n), (kd_3^n, w_3^n), \ldots, (kd_n^n, w_n^n) \rangle_{seq_n}^T \end{array} \right\}$$

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 $Content_{(x_j)}$ and each sequence seq_i in V_u^t. The technical details of vector alignment are described in the **APPENDIX** section, and we use the vector similarity $Sim_{(Content_{x_j}, seq_i)}$ to filter recordings. That is to say, if $Sim_{(Content_{x_j}, seq_i)}$ smaller than a given threshold ∂, then seq_i will not be used in the modeling process. Moreover, we should use the calculated $Sim_{(Content_{x_j}, seq_i)}$ as the weight to revise the TF-IDF weight corresponding to seq_i as:

$$seq_i.\{\vec{w}\} = Sim_{(Content_{x_j}, seq_i)} * seq_i.\{\vec{w}\}.$$

Finally, we will obtain a behavior recordings matrix S_u based on the $Content_{(x_j)}$, and the definition is shown as Eq. (5).

$$S_u = \left[\begin{array}{l} < (kd_1, w_1), timestamp_1 >_u \\ < (kd_2, w_2), timestamp_2 >_u \\ \cdots \\ < (kd_n, w_n), timestamp_n >_u \end{array} \right], \quad (5)$$

where S_u denotes a formatted behavior recordings of user u, and $timestamp_k$ 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 Eq. (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 Eq. (6).

$$tif_i = \frac{\ln\left(|h_i - h_{\text{now}}|^{-1}\right)}{\sum_{kd_j \in K_u} \ln\left(|h_j - h_{\text{now}}|^{-1}\right) + 1}, \quad (6)$$

As in Eq. (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 redefine as $\beta_i = (tif_i * TF - IDF_{(t_i)})$, which can be described in detail as Eq. (7).

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

Therefore, the user perference 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 rate 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. Table 1 shows a sample of feedback data.

TABLE I: A sample of feedback data

Items	Click rate	Browse duration(s)	Tags
Item-1	0.61	63	Health, Science
Item-2	0.38	126	Sports, Politics
Item-3	0.43	77	Entertainment
...	...	...	...
Item-n	0.53	88	Finance, Economics

Step 2: *Classifying feedback data*. After obtaining standardized feedback data, the next step is to classify these data according to recommendation effect. In this work, we define $Rindex$ to indicate the user's interest in recommendations based on the feedback data, which can be calculated as Eq. (8).

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

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, Here, $S_{(f_i)}$ is used to represent the score value of f_i after digitization and normalization.

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:

- 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.
- 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 paper.
- 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 $(\beta_i - \mu)$.

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**
    $dn_i \overset{send}{\rightarrow} positiveLibrary$;
  end if
  if $(Rindex_i \leq \gamma)$ **then**
    $dn_i \overset{send}{\rightarrow} negativeLibrary$;
  end if
end for
return true;

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 work, the collaborative filtering 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

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

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

- Calculate the prediction rating value $p_{u_t,\hat{\theta}}$ of user u_t for an unrated item $\hat{\theta}$ as Eq. (9):

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

where $r\rho_{t,i}$ denotes the similarity between user u_t and user u_i, which can be calculated as Eq. (4). Then, $p_{i,\hat{\theta}}$ 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 study, we use the *content filtering* method as one of the important components of our scheme. We can briefly introduce the process as follows:

- Let kd_i be the i-th 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 $Content_{(x_j)} = \{(kd_i, w_{1j}), (kd_2, w_{2j}), \ldots, (kd_n, w_{nj})\}$.
- 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 4.2 to build preference model as $PM_{(u)} = \{(kd_1, \beta_1), (kd_2, \beta_2), \ldots, (kd_n, \beta_n)\}$.
- In general, we should map $PM_{(u)}$ and $Content_{(x_j)}$ 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 work, 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.
- After completing the attribute mapping, $PM_{(u)}$ and $Content_{(x_j)}$ will be unified into the same *VSM*. Then, we should calculate the similarity between the content vector $Content_{(x_j)}$ and the user preference vector $PM_{(u)}$ as Eq. (10).

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

where the cosine similarity is used to measure the degree of compliance with user preferences, and w_{ij} is the weight of the i-th attribute in the vector $Content_{(x_j)}$.

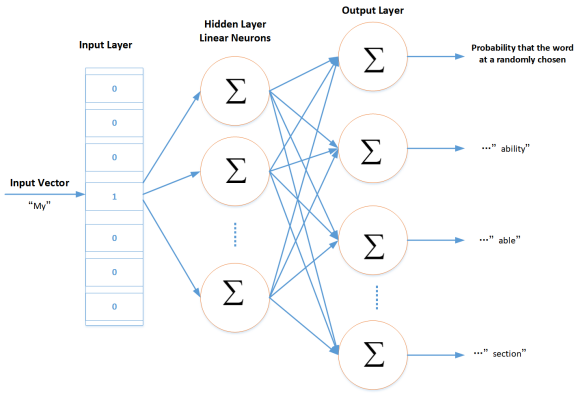


Fig. 2: Skip-gram model.

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.

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 value based on collaborative filtering 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 $P_{(CB_u)} = \{\rho_{u,x_1}, \rho_{u,x_2}, \ldots, \rho_{u,x_n}\}$. 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 ρ_{u,x_i}.

Logistic regression is used to aggregate the results of recommendation both collaborative filtering and content-based. Then $P_{(CF_u)}$ and $P_{(CB_u)}$ should be as parameters of Eq. (11).

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

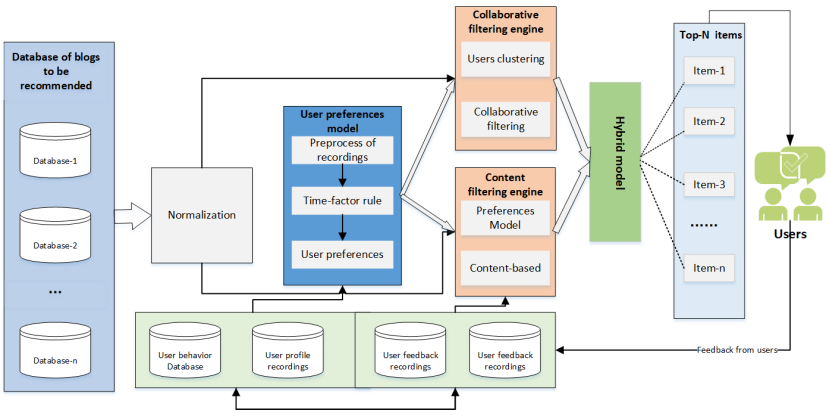


Fig. 3: Hybrid model.

where $h_\theta(X)$ can be defined as Eq. (12).

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

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 $P_{(CB_u)} = \{\rho_{u,x_1}, \rho_{u,x_2}, \ldots, \rho_{u,x_n}\}$, 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.

Step1: Using the above collaborative filtering in subsection 4.3 to obtain the recommendation items $D_r = \{d_1, d_2, \ldots, d_n\}$ and the corresponding prediction rating value $P_{(CF_u)} = \{p_{u,1}, p_{u,2}, \ldots, p_{u,n}\}$.

Step2: Using the above content-based model defined in subsection 4.4 to obtain the recommendation items prediction rating value of D_r as $P_{(CB_u)} = \{\rho_{u,x_1}, \rho_{u,x_2}, \ldots, \rho_{u,x_n}\}$.

Step3: $P_{(CF_u)} = \{p_{u,1}, p_{u,2}, \ldots, p_{u,n}\}$ and $P_{(CB_u)} = \{\rho_{u,x_1}, \rho_{u,x_2}, \ldots, \rho_{u,x_n}\}$ are selected as input parameters of Eq. (12). Then the comprehensive rating $FM = \{\omega_1, \omega_2, ..., \omega_n\}$ would be generated.

Step4: $FM = \{\omega_1, \omega_2, ..., \omega_n\}$ 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 subsection 4.2) to achieve better recommendations.

V. Experiments and Results

In this section, we conduct experiments for evaluating the effectiveness of our scheme for the top-N recommendation task. The experiments are designed to compare the performance of our scheme with that of other state-of-the-art related methods, and to study the impact of relevant parameters on the experimental results.

A. Data preparation

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

- **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 paper, we mainly focus on VLE (Virtual Learning Environment) data, which show learner preference in choosing learning materials.
- **MovieLens-Latest** dataset is collected as part of the GroupLens Research Project of the University of Minnesota (and are available online at https://grouplens.org/datasets/movielens/). This dataset consists of 27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users.
- **Book-Crossing** dataset is collected by Cai-Nicolas Ziegler from the Book-Crossing community (and are available online at https://grouplens.org/datasets/book-crossing/). 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. To avoid this issue, we remove records where at least one of the features has more than 5 standard deviations. For the Book-Crossing dataset, we remove about 8,576 records from its Bx-Book-Ratings.csv. Then, the statistics of these three datasets are shown in Table 2.

B. Evaluation metrics

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

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

TABLE II: 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 sparsity	0.076%	0.016%	0.012%

where *Precision* is the ratio of the correctly judged existent propagation relations to all the judged existent propagation relations.

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

where *Recall* is the ratio of the correctly judged existent propagation relations to all the existent propagation relations in system.

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

where *F-score* is the harmonic mean (average) of the precision and recall. Hence, F-score will be a better measure when precision and recall are sometimes contradictory. The meanings of TP, FP and FN are described in Table 3.

TABLE III: 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

C. Evaluation baselines

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

- **PRMR** [34]: This method is proposed to improve the efficiency of Collaborative Filtering (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, such that the dimension of the original user-item matrix can be significantly reduced. Thus, the method can significantly reduce the time complexity of Collaborative Filtering (CF).
- **AROLS** [35]: This work 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 creates learner clusters according to their online learning styles, and applies Collaborative Filtering (CF) and association rule mining to extract the preferences and behavioral patterns of each cluster.
- **HRBRM** [36]: The most important achievement of this study is to present a novel approach in hybrid recommendation systems, which identifies the user similarity neighborhood from implicit information being collected in a discussion group. In the proposed system, initially the association rules mining technique is applied to discover similar users, and then the related posts are recommended to them. Besides, to locate the semantically related concepts Word Sense Disambiguation strategy based on WordNet lexical database is exploited.
- **HRSRL** [37]: This work proposes a hybrid recommendation system, combining content-based and collaborative filtering together 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.

Note that for all comparison methods, we tune the hyper-parameters carefully according to corresponding references to ensure that each method achieves its performance for fair comparison. For each employed dataset, we randomly choose 80% of the data in datasets for the training process and the others for the testing process.

D. Experimental results

In our experiments, the interval threshold parameters in Algorithm 1 are set by $\lambda = 0.35$ and $\gamma = 0.46$, then the impact of parameters will be studied in Section 5.5. For performance comparison, we report the recommendation precision and recall of different methods over the three datasets in Table 4. It is important to note that all the results in Table 4 are the mean performance of each experiment for 5 times.

From the results in Table 4, 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. Overall, our scheme performs better than the four baselines by at least 7.7% and 8.3% for all datasets on metrics of precision and recall, respectively. From Table 4, we can see that our scheme can perform better than PRMR about 16.1% and 11.2% on metrics of precision and recall, respectively. PRMR is used to improve the efficiency of collaborative filtering 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 collaborative filtering (CF) and association rule mining to extract the preferences without using any user feedback mechanism, then this may lead its precision and recall are still lower than that of our scheme about 12.1% and 11.4% respectively.

Second, from the results, we find that the hybrid methods perform better than traditional recommendation algorithms. For instance, HRBRM performs better than PRMR and AROLS by about 8.7% and 7.3% in terms of recall. HRSRL also perform better than PRMR and AROLS by about 9.6% and 8.5% on the metric of precision, respectively. This is because these hybrid schemes can take advantage of collaborative filtering 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 10.2% on metrics of precision and recall, respectively. 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. 4. For our proposed scheme, F1-score changes slightly with the number of recommendation items, while the baselines also change around certain values. As shown in Fig.4, our scheme achieves the best performance for F1-score on all the three datasets. Specifically, the average F1-score of our scheme is about 9.6%, 8.8%, 6.7% and 7.1% 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 collaborative filtering 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.

E. Impact of parameters

In this section, we analyze how the parameters affect the performance of our proposed scheme on all the employed datasets (i.e., OULAD, MovieLens-Latest, 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. From the results in Fig. 5, we can see that our scheme can achieve satisfying performance when the values of λ and γ are not too big or small. Specifically, we first evaluate the impact of parameter λ on the recommendations, here, we set $\gamma = 0.25$ and Top-N=25 (i.e., the number of recommendation items). We can observe that the 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 extent, the corresponding F1-score decreases. This may be the accuracy of the feedback library decreases. Then, based on the same principles, 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), at the beginning, the recommendation effect will increase with γ due to the increase of the feedback recordings. When γ increases to a certain extent, the recommendation effect will decrease accordingly, this is because the accuracy of the feedback library decreases as γ increases to a certain extent.

VI. Conclusions

In this paper, a hybrid recommendation scheme is proposed. We propose the definition of the time impact factor and apply it to the user preference modeling to ensure that the change of user interest can be captured. A user feedback mechanism is proposed in this work, we use the user's feedback information on recommended items to establish positive and negative feedback libraries, and filter the recommendation results based on the feedback mechanism. To improve the efficiency of traditional collaborative filtering, spectral clustering algorithm is utilized. Finally, the logistic regression algorithm is utilized to aggregate the results of our proposed collaborative filtering and content-based. Then the proposed feedback mechanism is used to filter the result of comprehensive recommendations. Experiments show that our proposed scheme achieves the best recommendation efficiency compared with baselines.

Appendix

In this work, the defined interest vector is essentially the weight vector of aseries of keywords(features), we consider that the inner product operation needs the corresponding feature alignment. Hence, we put the mapping operation of vector space when we need to do the inner product operation, so that the PM_i can retain more information. For instance, we assume that Content $_x = \{(f_1^x, w_1), (f_2^x, w_2), \ldots, (f_n^x, w_n)\}$ is the one of target items to the user u_i with $PM_i = \{(f_1^i, w_1), (f_2^i, w_2), (f_4^i, w_4), \ldots, (f_n^i, w_n)\}$. Here, we should construct a target feature vector based on the $Content_x$ as $F_{tar}^x = \{f_1^x, f_2^x, \ldots, f_n^x\}$. Therefore, the alignment of the vector attributes should be based on F_{tar}^x, and the following steps should be adopted.

Step 1: We should extract the origianl features $F_o^i = \{f_1^i, f_2^i, f_4^i, \ldots, f_n^i\}$ from PM_i, then we use the *wordnet* to calculte the similariy between F_o^i and F_{tar}^x as shown in **Alg. 2** and Fig. 6.

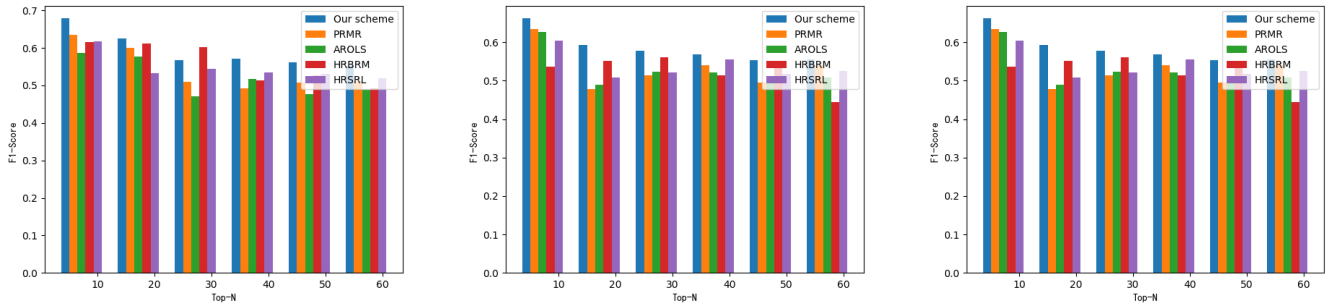
Algorithm 2 Building synonyms vector

```
for f_j^i in F_O^i do
    for f_y^x in F_tar^x do
        wordsim=Call sim.getSimilarity(f_j^i, f_y^x);
        If wordsim > maxsim:
            maxsim = wordim; memindex=< y, j >;
        End for
        If maxsim >= ∂:
            f_g^i=f_(memindex.y)^x;
            w_g^i=PM(memindex.j);
            setVector(f_g^i ⇒ w_g^i);
End for
return true;
```

Step 2: As mentioned above, we can filter out all the feature words (synonym) similar to those in F_{tar}^x. Besides, we should consider whether the non synonyms $f_k^i \in F_o^i$ and F_{tar}^x has

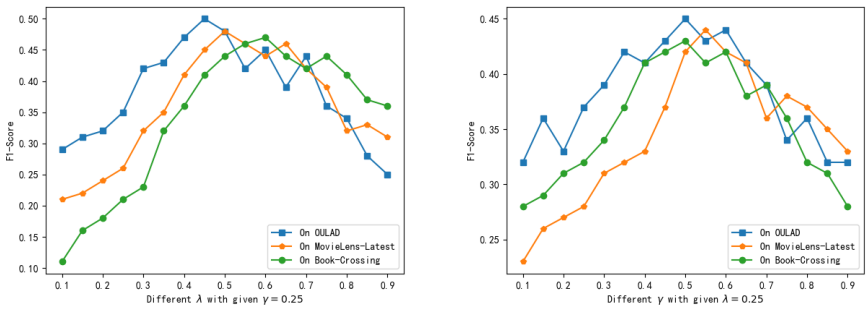
TABLE IV: The evaluation of recommendation quality in terms of precision, recall.

Top-N	DataSet	Ours		PRMR		AROLS		HRBRM		HRSRL	
		Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
10	OULAD	0.705	0.551	0.703	0.579	0.701	0.505	0.701	0.543	0.711	0.531
	MovieLens-Latest	0.503	0.523	0.691	0.586	0.521	0.495	0.601	0.484	0.723	0.521
	Book-Crossing	0.614	0.501	0.561	0.576	0.631	0.452	0.568	0.516	0.689	0.483
20	OULAD	0.713	0.452	0.623	0.581	0.661	0.512	0.681	0.562	0.691	0.489
	MovieLens-Latest	0.484	0.504	0.423	0.554	0.501	0.479	0.522	0.582	0.541	0.481
	Book-Crossing	0.632	0.494	0.608	0.536	0.621	0.465	0.616	0.542	0.556	0.456
30	OULAD	0.693	0.464	0.523	0.497	0.594	0.473	0.671	0.546	0.624	0.483
	MovieLens-Latest	0.527	0.479	0.596	0.602	0.576	0.481	0.584	0.532	0.531	0.515
	Book-Crossing	0.618	0.508	0.579	0.472	0.582	0.511	0.605	0.519	0.547	0.462
40	OULAD	0.681	0.494	0.547	0.447	0.547	0.491	0.573	0.465	0.657	0.451
	MovieLens-Latest	0.604	0.521	0.611	0.503	0.501	0.545	0.633	0.503	0.601	0.516
	Book-Crossing	0.584	0.494	0.544	0.576	0.451	0.483	0.551	0.483	0.573	0.443
50	OULAD	0.673	0.482	0.565	0.462	0.501	0.456	0.561	0.497	0.603	0.475
	MovieLens-Latest	0.623	0.501	0.543	0.456	0.492	0.495	0.526	0.432	0.562	0.481
	Book-Crossing	0.601	0.531	0.523	0.481	0.513	0.531	0.558	0.504	0.558	0.476
60	OULAD	0.567	0.484	0.563	0.463	0.493	0.485	0.529	0.462	0.535	0.501
	MovieLens-Latest	0.645	0.489	0.619	0.482	0.505	0.507	0.482	0.447	0.612	0.462
	Book-Crossing	0.586	0.472	0.541	0.447	0.489	0.453	0.476	0.526	0.501	0.469



(a) F1-score on OULAD (b) F1-score on MovieLens-Latest (c) F1-score on Book-Crossing

Fig. 4: Results of F1-score on all used datasets.



(a) Parameter study on λ. (b) Parameter study on γ.

Fig. 5: Parameters study on γ and λ.

certain semantic similarity. To specific, we used a trained Skip-gram model to generate embeddings. The process can be briefly described in Fig. 7.

Step 3: Based on the Step 2, we could calculate the semantic similarity of the non synonyms $f_k^i \in F_o^i$ and $f_j^x \in F_{tar}^x$ by using the vector similarity as $\mathrm{Sim}_{k,j} = \frac{EV_{(f_k^i)} \bullet EV_{(f_j^x)}}{\left| EV_{(f_k^i)} \| EV_{(f_j^x)} \right|}$. Also, the obtained $Sim_{k,j}$ could be used as weights of aggregate operations.

Step 4: We should aggregate all features in F_o^i to generate a new interest vector that matches the target vector F_{tar}^x in both length and attributes. To specific, we should calculate the semantic similarity of non synonyms $f_k^i \in F_o^i$ and each element in F_{tar}^x, then obtain a set of semantic similarity weight coefficients $W_{\text{sem}} = \{w_1^s, w_2^s, \ldots, w_n^s\}$. In fact, we

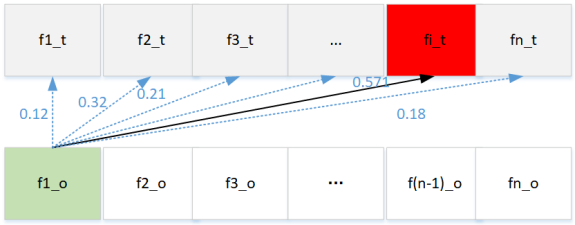


Fig. 6: Similarity calculation based on wordnet.

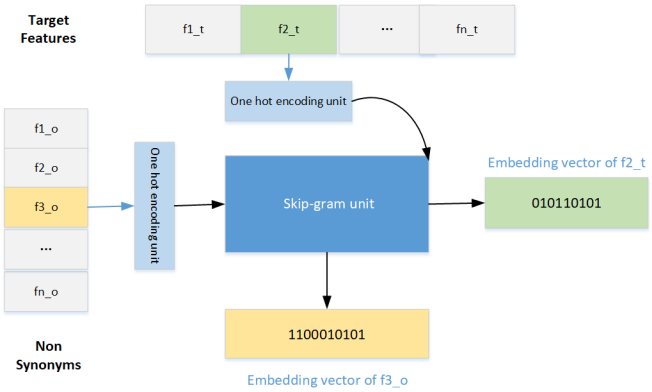


Fig. 7: Word embedding.

can calculate the weight of our new interest vector with the weighted sum operations, and key steps are shown in **Alg. 3**.

Algorithm 3 Aggregate vector weight

1: **for** non synonyms $f_k^i \in F_o^i$ **do**
2: $s_k^i = \text{getWeightFromPM}_i\left(f_k^i\right)$;
3: **for** f_j^x in F_{tar}^x **do**
4: $w_{k,j} = \frac{EV_{\left(f_k^i\right)} \bullet EV_{\left(f_j^x\right)}}{\left|EV_{\left(f_k^i\right)} \| EV_{\left(f_j^x\right)}\right|}$;
5: f_{syn}^i= Call **Alg .1**&getSynonyms(f_j^x);
6: $nweight = getWeightFromPM_i\left(f_{syn}^i\right) + w_{k,j} * s_k^i$;
7: $setWeightOfVector_i\left(f_{syn}^i, nweight\right)$;
8: **End for**
9: **End for**
10: **return** true;

Note that if there are multiple synonyms, their corresponding weights are added directly and the feature attributes should be merged.

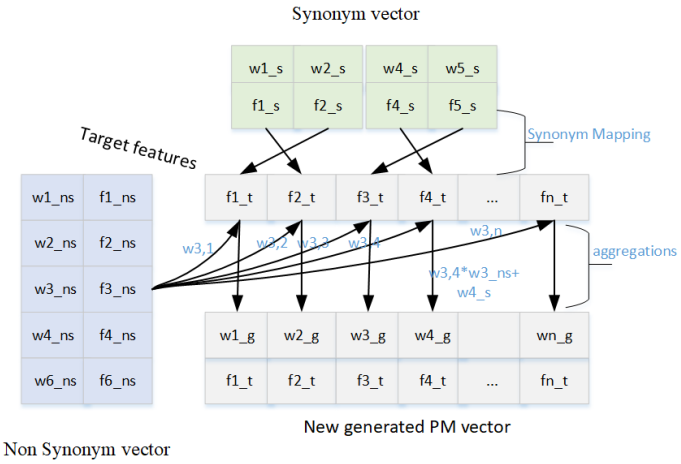


Fig. 8: Vector alignment calculation.

References

[1] Analysis of popular Internet application recommendation algorithm., 2019. [Online]. Available: https://zhuanlan.zhihu.com/p/88815475

[2] Peizhen Bai, Yan Ge, Fangling Liu, and Haiping Lu. Joint interaction with context operation for collaborative filtering. *Pattern Recognition*, 88:729–738, 2019.

[3] Virane Gautam, Upasana Sharma, Mayank Sharma, and Sunil Kumar Khatri. A noval collaborative filtering technique approach for recommender system. In *2019 Amity International Conference on Artificial Intelligence (AICAI)*, pages 151–155. IEEE, 2019.

[4] Ashutosh Lokhande and Pooja Jain. Hybrid collaborative filtering model using hierarchical clustering and pca. *Available at SSRN 3365525*, 2019.

[5] Feng Liu and Wei-wei Guo. Research on recommendation system algorithm based on deep learning mode in grid environment. In *2019 International Conference on Robots & Intelligent System (ICRIS)*, pages 250–253. IEEE, 2019.

[6] Wei Peng and Baogui Xin. Spmf: A social trust and preference segmentation-based matrix factorization recommendation algorithm. *arXiv preprint arXiv:1903.04489*, 2019.

[7] Dan Wu. Music personalized recommendation system based on hybrid filtration. In *2019 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS)*, pages 430–433. IEEE, 2019.

[8] Jianli Zhao, Xijiao Geng, Jiehan Zhou, Qiuxia Sun, Yu Xiao, Zeli Zhang, and Zhengbin Fu. Attribute mapping and autoencoder neural network based matrix factorization initialization for recommendation systems. *Knowledge-Based Systems*, 166:132–139, 2019.

[9] Hashem Parvin, Parham Moradi, and Shahrokh Esmaeili. Tcfaco: Trust-aware collaborative filtering method based on ant colony optimization. *Expert Systems with Applications*, 118:152–168, 2019.

[10] Shulin Cheng and Wanyan Wang. Rating prediction algorithm based on user time-sensitivity. *Information*, 11(1):4, 2020.

[11] Yu Li, Liu Lu, and Li Xuefeng. A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in e-commerce. *Expert systems with applications*, 28(1):67–77, 2005.

[12] R Logesh and V Subramaniyaswamy. Exploring hybrid recommender systems for personalized travel applications. In *Cognitive informatics and soft computing*, pages 535–544. Springer, 2019.

[13] Monali Gandhi and Sonali Gandhi. An enhanced approach for tourism recommendation system using hybrid filtering and association rule mining. *Asian Journal For Convergence In Technology (AJCT)*, 2019.

[14] Lai-Ying Leong, Teck-Soon Hew, Keng-Boon Ooi, and Alain Yee-Loong Chong. Predicting the antecedents of trust in social commerce–a hybrid structural equation modeling with neural network approach. *Journal of Business Research*, 110:24–40, 2020.

[15] Xiao-Lin Zheng, Chao-Chao Chen, Jui-Long Hung, Wu He, Fu-Xing Hong, and Zhen Lin. A hybrid trust-based recommender system for online communities of practice. *IEEE Transactions on Learning Technologies*, 8(4):345–356, 2015.

[16] H Lee and J Um. A study on the context-aware hybrid bayesian recommender system on the mobile devices. *IAENG International Journal of Computer Science*, 45(1):40–45, 2018.

[17] Amirali Salehi-Abari and Craig Boutilier. Preference-oriented social networks: Group recommendation and inference. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 35–42. ACM, 2015.

[18] Xiaoyan Wang, Lifeng Sun, Zhi Wang, and Da Meng. Group recommendation using external followee for social tv. In *2012 IEEE International Conference on Multimedia and Expo*, pages 37–42. IEEE, 2012.

[19] Lifeng Sun, Xiaoyan Wang, Zhi Wang, Hong Zhao, and Wenwu Zhu. Social-aware video recommendation for online social groups. *IEEE Transactions on Multimedia*, 19(3):609–618, 2016.

[20] Huizhi Liang, Yue Xu, Dian Tjondronegoro, and Peter Christen. Time-aware topic recommendation based on micro-blogs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1657–1661. ACM, 2012.

[21] Shipra Goel, Muskan Banthia, and Adwitiya Sinha. Modeling recommendation system for real time analysis of social media dynamics. In

2018 Eleventh International Conference on Contemporary Computing (IC3), pages 1–5. IEEE, 2018.

[22] Enrico Palumbo, Giuseppe Rizzo, and Raphaël Troncy. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 32–36. ACM, 2017.

[23] Luis M De Campos, Juan M Fernández-Luna, Juan F Huete, and Miguel A Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, 2010.

[24] Chun Lu, Philippe Laublet, and Milan Stankovic. Travel attractions recommendation with knowledge graphs. In *European Knowledge Acquisition Workshop*, pages 416–431. Springer, 2016.

[25] Rose Catherine and William Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 325–332. ACM, 2016.

[26] H Wang, N Wang, and DY Yeung. Collaborative deep learning for recommender systems, 1235–1244. *Google Scholar Google Scholar Digital Library Digital Library*, 2014.

[27] Kerry-Louise Skillen, Liming Chen, Chris D Nugent, Mark P Donnelly, William Burns, and Ivar Solheim. Ontological user modelling and semantic rule-based reasoning for personalisation of help-on-demand services in pervasive environments. *Future Generation Computer Systems*, 34:97–109, 2014.

[28] Cataldo Musto. Enhanced vector space models for content-based recommender systems. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 361–364. ACM, 2010.

[29] Shangsong Liang, Xiangliang Zhang, Zhaochun Ren, and Evangelos Kanoulas. Dynamic embeddings for user profiling in twitter. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1764–1773. ACM, 2018.

[30] Lu Yu, Chuang Liu, and Zike Zhang. Multi-linear interactive matrix factorization. *Knowledge Based Systems*, 85:307–315, 2015.

[31] Open University Learning Analytics dataset., 2017. [Online]. Available: https://www.nature.com/articles/sdata2017171/

[32] Lu Yu, Junming Huang, Ge Zhou, Chuang Liu, and Zike Zhang. Tiirec: A tensor approach for tag-driven item recommendation with sparse user generated content. *Information Sciences*, 411:122–135, 2017.

[33] Jarvan Law, Hankz Hankui Zhuo, Junhua He, and Erhu Rong. Ltsg: Latent topical skip-gram for mutually learning topic model and vector representations. *arXiv: Computation and Language*, 2017.

[34] Jiang Zhang, Yufeng Wang, Zhiyuan Yuan, and Qun Jin. Personalized real-time movie recommendation system: Practical prototype and evaluation. *Tsinghua Science & Technology*, 25(2):180–191, 2020.

[35] Hui Chen, Chuantao Yin, Rumei Li, Wenge Rong, Zhang Xiong, and Bertrand David. Enhanced learning resource recommendation based on online learning style model. *Tsinghua Science & Technology*, 25(3):348–356, 2020.

[36] Ahmad A Kardan and Mahnaz Ebrahimi. A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups. *Information Sciences*, 219:93–110, 2013.

[37] Shuo Yang, Mohammed Korayem, Khalifeh Aljadda, Trey Grainger, and Sriraam Natarajan. Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive statistical relational learning approach. *Knowledge Based Systems*, 136:37–45, 2017.

Hongzhi Li received the M.S. degree from Wuhan University of Technology of computer science and engineering. He is currently pursuing the Ph.D. degree at Shanghai Maritime University. His current research interests include recommendation systems and information security.

Dezhi Han received the BS degree from Hefei University of Technology, Hefei, China, the MS degree and PhD degree from Huazhong University of Science and Technology, Wuhan, China. He is currently a professor of computer science and engineering at Shanghai Maritime University. His specific interests include storage architecture, cloud computing, cloud computing security and cloud storage security technology.