Recommendation Algorithm Integrating Item Bias and User Preferences (Postprint)

Authors: Cheng Lei, Maoting Gao

Date: 2018-08-13T00:00:00+00:00

Abstract

To address the issues of item bias and user preferences arising from the mutual influence of correlation factors between items and users in collaborative filtering recommendation, a recommendation algorithm integrating item bias and user preferences is proposed. The algorithm first conducts clustering processing, which includes generating item clusters through LDA topic modeling and user clusters through K-means clustering. Subsequently, item bias scores are generated sequentially based on constraints from both item clusters and user clusters, while user preference scores are obtained through probability transition grounded on user-item ratings and item types. Finally, prediction ratings are produced via linear weighting of item bias scores and user preference scores, based on the mean of existing ratings within the item cluster. Comparative experiments demonstrate that the proposed algorithm can yield reasonable recommendations according to different neighbors and enhance recommendation accuracy.

Full Text

Preamble

Title: Recommendation Algorithm Combining Item Deviation and User Preference

Authors: Cheng Lei, Gao Maoting
(College of Information Engineering, Shanghai Maritime University, Shanghai 201306, China)

Abstract: Collaborative filtering recommendation suffers from item deviation and user preference problems due to the mutual influence of correlation factors between items and users. To address this issue, this paper proposes a recommendation algorithm that integrates item deviation and user preference. The algorithm first performs clustering, including LDA topic modeling to generate

item clusters and K-means clustering to generate user clusters. It then generates an item deviation score based on constraints from both item and user clusters, while simultaneously deriving a user preference score through probability transfer based on user-item ratings and item types. Finally, it linearly weights the item deviation score and user preference score to generate a predicted rating, using the mean of existing ratings within the item cluster as a baseline. Comparative experiments demonstrate that the proposed algorithm can produce reasonable recommendations based on different neighbors and improve recommendation accuracy.

Keywords: collaborative filtering; topic modeling; clustering; item deviation; user preference

0 Introduction

Recommendation systems serve as important tools to help users quickly select relevant information and are increasingly employed by e-commerce and social networking sites to enhance user experience. Recommendation algorithms primarily include collaborative filtering-based, content-based, and tag-based approaches [**?**]. As a commonly used technique, collaborative filtering predicts ratings based on user-item rating information. However, several limitations hinder accuracy improvements. On one hand, due to mutual influences among correlation factors between items, item deviation problems exist in practice. On the other hand, insufficient consideration is given to preference factors that different users have toward various item types.

To address these issues, Shi et al. [**?**] proposed a novel probabilistic topic model that introduces a penalty term for dissimilarity between two items during similarity computation, reducing the impact of irrelevant items on accuracy. Qiao et al. [**?**] proposed a recommendation algorithm combining user attributes and item content, utilizing the LDA (Latent Dirichlet Allocation) model to perform topic analysis on user attributes and item content, thereby 挖掘 ing user preferences for items. Zhao et al. [**?**] introduced a recommendation algorithm based on feature transfer and probabilistic matrix factorization, integrating a trust matrix into the rating matrix so that user ratings are only influenced by their own attributes and trusted individuals, filtering out irrelevant users. Zhou et al. [**?**] proposed an LDA model for evaluating collaborative filtering, which performs collaborative filtering by adding user rating information to the topic model and provides reasonable recommendations based on user preferences for items. Yuan et al. [**?**] proposed an item-based collaborative filtering algorithm that incorporates item classification and K-nearest neighbors into the Slope One algorithm, filtering out irrelevant items. Liu et al. [**?**] proposed a matrix factorization algorithm based on user preferences, calculating user preferences through the user-item rating matrix and the item attribute matrix obtained from matrix factorization, thereby improving prediction accuracy.

These studies have considered the impact of item deviation or user preference factors to some extent and achieved good results. Building upon this foundation, we propose a recommendation algorithm that fuses item deviation and user preference (Item Deviation and User Preference Combination Filtering, IUCF). Through clustering processing of items and users separately, the algorithm leverages nearest-neighbor collaborative filtering and probability transfer to 挖掘 item deviation and user preference, integrating both into the final predicted rating for recommendation.

---

1 Related Research

1.1 User-Based Collaborative Filtering Algorithm

The user-based collaborative filtering algorithm consists of the following steps: (a) calculate similarity between users based on the user-item rating matrix; (b) select neighbor users according to similarity; and (c) generate predicted ratings based on neighbor users. Similarity is computed using the Pearson correlation coefficient [**?**]:

$$\mathrm{Sim}(a,b) = \frac{\sum_{n\in I_{ab}}(r_{an}-\bar{r}_a)(r_{bn}-\bar{r}_b)}{\sqrt{\sum_{n\in I_{ab}}(r_{an}-\bar{r}_a)^2}\sqrt{\sum_{n\in I_{ab}}(r_{bn}-\bar{r}_b)^2}}$$

where I_{ab} represents the set of items co-rated by user a and user b, r_{an} and r_{bn} denote the actual ratings given by user a and user b to item n, and $\bar{r}_a$ and $\bar{r}_b$ represent the average ratings of user a and user b, respectively.

1.2 LDA Topic Model

The LDA topic model [**?**] first uses Dirichlet probability distribution to set the latent probabilities of documents, then employs a sampling algorithm to estimate the document-topic probability distribution and topic-word probability distribution. The sampling algorithm adopts Gibbs sampling [**?**]:

$$p(z_i = k|z_{\neg i}, w) \propto \frac{n_{m,k}^{\neg i}+\alpha_k}{\sum_{k=1}^{K}(n_{m,k}^{\neg i}+\alpha_k)} \cdot \frac{n_{k,t}^{\neg i}+\beta_t}{\sum_{t=1}^{V}(n_{k,t}^{\neg i}+\beta_t)}$$

where $z_{\neg i}$ denotes the topic distribution calculated from the sequence of other words in document set D excluding the i-th word, w_i represents the word corresponding to the i-th word in corpus D; $n_{m,k}^{\neg i}$ indicates the count of words belonging to topic k in document m excluding the i-th word; $n_{k,t}^{\neg i}$ indicates the count of word t in topic k excluding the i-th word; and α and β represent the Dirichlet prior parameters for document-topic distribution and topic-word distribution, respectively.

1.3 Problem Description and Analysis

During the rating prediction process, due to the mutual influence of correlation factors between items, item deviation problems exist in practice. For example, when calculating ratings for action movies, ratings from many irrelevant genres also participate in the computation, causing predicted ratings to deviate from actual values. Similarly, when selecting neighbors, the search is often conducted across the entire user set without fully considering users' intrinsic attributes, making it impossible to accurately reflect preferences that different users have for different item types. For instance, hobbies vary significantly across age groups, and users with seemingly similar ratings may actually have substantial differences. Furthermore, user preferences also influence their ratings for items. For example, when a movie contains both action genres that a user likes and horror genres that they dislike, the user's rating will be affected by their personal preferences. Therefore, it is necessary to accurately 挖掘 item deviation and user preferences to correct the final predicted ratings.

Regarding item deviation 挖掘, 文献 [6] filtered item deviation from both user and item perspectives before hybrid weighting, but this approach involves high computational complexity. 文献 [11] used linear weighting of user trust and similarity as the basis for neighbor selection, yet still failed to consider the intrinsic attributes of items during calculation. Regarding user preference 挖掘, 文献 [7] 挖掘 ed user preferences through matrix factorization algorithms, but matrix factorization lacks interpretability. 文献 [12] simply used the mean rating of items of a certain type to represent the type' s rating, resulting in low discriminability of user preferences.

Therefore, in the IUCF algorithm, we fully consider the intrinsic attributes of items and users. By clustering item types and user attributes, we make item deviation more accurate and reliable. By 统计 the proportion of times item types are tagged and combining this with the user-item rating matrix, we enable frequently occurring types to receive higher scores, thereby making the obtained user preferences reliable.

---

2 Recommendation Algorithm Combining Item Deviation and User Preference

The IUCF algorithm incorporates two components: generating an item deviation score based on item and user clusters, and generating a user preference score based on item types. It then uses the mean of existing ratings in the target user' s item cluster as a baseline and linearly weights these two parts to generate the final predicted rating. The algorithm model is shown in Figure 1 [Figure 1: see original paper].

2.1 Clustering Processing

To accurately reflect item deviation, it is necessary to consider both item types and user attributes in the item deviation calculation. Simultaneously, to accurately 挖掘 user preferences, we must consider users' preference degrees for different item types. Therefore, by clustering item types and user attributes to generate item clusters and user clusters, we calculate item deviation and user preferences to improve recommendation accuracy.

2.1.1 LDA Topic Modeling for Item Cluster Generation LDA topic modeling is a document topic generation model that can identify latent topic information in a corpus. In this model, each document is considered to be composed of many topics, and each topic is composed of many words. We treat all item types as documents and types as words to find the topics where each word resides. Let $I = \{I_1, I_2, \ldots, I_n\}$ be the item set, where I_n represents the n-th item. Define the item type $i_m = (i_{m1}, i_{m2}, \ldots, i_{mn})$, where i_{mn} represents the n-th type of item I_m. For example, when $i_m = (\text{Action}, \text{Drama}, \text{War})$, it indicates that item I_m has types Action, Drama, and War.

We perform LDA topic modeling on the item set using the Gibbs sampling method to obtain the item-type topic distribution $\hat{Z}_{t\times m}$ and the topic-type probability matrix $\hat{Z}_{n\times m}$, where the number of topics needs to be determined based on the algorithm' s accuracy after LDA clustering.

In $\hat{Z}_{t\times m}$, each item I_n' s type i_m has a topic label, in the form $\hat{Z}_{n\times m} = (z_{i1}, z_{i2}, \ldots, z_{im})$, where z_{im} represents the topic label corresponding to type i_m. For example, Action : 3 indicates that type Action belongs to topic 3.

Based on $\hat{Z}_{n\times m}$, we establish the item-topic membership matrix $\hat{I}_{n\times t}$, where $\hat{I}_{nt} = 1$ when item I_n' s type belongs to topic t, otherwise $\hat{I}_{nt} = 0$, as shown in Equation (3):

$$\hat{I}_{nt} = \begin{cases} 1 & \text{if type } i_m \text{ of item } I_n \text{ belongs to topic } t \\ 0 & \text{otherwise} \end{cases}$$

From any row of the item-topic membership matrix, we can obtain the item topic cluster $CN_n = \{i_n | \hat{I}_{nt} = 1, t \in [1, t]\}$. From any column, we can obtain the topic item cluster $CT_t = \{i_n | \hat{I}_{nt} = 1, n \in [1, n]\}$, as shown in Equations (4) and (5):

$$CN_n = \{i_n | \hat{I}_{nt} = 1, t \in [1, t]\}$$

$$CT_t = \{i_n | \hat{I}_{nt} = 1, n \in [1, n]\}$$

where CN_n represents the set of topics to which item I_n belongs, CT_t represents the set of items contained in topic t, t represents the topic number, and n represents the item number.

For a target item I_n, we first find its corresponding CN_n, then find each topic' s corresponding CT_t, and finally perform a union operation on all CT_t to obtain the item cluster C_n where the target item resides, as shown in Equation (6):

$$C_n = \bigcup_{t \in CN_n} CT_t$$

2.1.2 K-means Clustering for User Cluster Generation In the current network environment, user attributes can be accurately obtained, including gender, age, and occupation. Define the user set as $U = \{U_1, U_2, \ldots, U_a\}$, and user attributes as $Q_i = (q_{i1}, q_{i2}, \ldots, q_{ik})$, where U_i represents the i-th user and q_{ik} represents the k-th basic attribute of user U_i. For example, when $Q_i = (\text{male}, 23, \text{teacher})$, it indicates that user U_i is male, 23 years old, and works as a teacher.

To perform K-means clustering, we need to preprocess users' basic information using digital encoding [1-9]. For gender, male and female are encoded as 1 and 2, respectively. For age, according to the five-group division method proposed in 文献 [13], user ages are divided into children (0-19), youth (20-39), middle-aged (40-59), senior (60-79), and elderly (80+), encoded sequentially as 1, 2, 3, 4, 5. For occupation, 文献 [14] applies the Pareto principle to information evaluation, which states that in any set of items, the most important ones account for only about 20%, while the remaining 80% are less significant. Similarly, we 统计 all user occupation types and the number of users in each occupation, rank occupations in descending order by user count, assign separate encodings to the top 20% of occupations, and group the remaining occupations into one category. For example, 统计 ing the occupation distribution in MovieLens reveals 21 occupations; we assign separate encodings to the top 4 occupations by user count and group the rest into one category, resulting in occupation encodings of 1, 2, 3, 4, 5. After digital encoding, user attributes are represented as numeric codes. For instance, $Q_i = (\text{male}, 23, \text{teacher})$ is represented as $Q_i = (1, 1, 5)$.

Since K-means clustering is an unsupervised learning method, the optimal number of clusters needs to be determined based on the sum of squares errors (SSE) across all clusters. We employ the elbow method [15], which selects the inflection point on the curve of SSE versus cluster count as the optimal number. After K-means clustering, we obtain the user cluster U_a where user U_i resides, as shown in Equation (7):

$$U_a = \{U_j | j \in [1, a], U_j \in Q_i\}$$

2.2 Calculating Item Deviation Score

We measure item deviation by the degree to which a target item' s ratings deviate from the item cluster' s mean. To accurately express the item deviation score, we primarily filter items and users through item clusters and user clusters, calculating the difference between each item' s rating and the mean of existing ratings in its item cluster. The calculation process is as follows:

a) **Calculate enhanced rating similarity within the item cluster.** Within item cluster C_n, we compute the enhanced rating similarity $\hat{\text{Sim}}(a,b)$ between users based on the user-item rating matrix, as shown in Equation (8):

$$\hat{\text{Sim}}(a,b) = \frac{\sum_{n \in I_{ab} \cap C_n} (r_{an} - \bar{r}_a^{C_n})(r_{bn} - \bar{r}_b^{C_n})}{\sqrt{\sum_{n \in I_{ab} \cap C_n} (r_{an} - \bar{r}_a^{C_n})^2} \sqrt{\sum_{n \in I_{ab} \cap C_n} (r_{bn} - \bar{r}_b^{C_n})^2}}$$

where $\bar{r}_a^{C_n}$ and $\bar{r}_b^{C_n}$ represent the means of existing ratings in C_n for items rated by users a and b, respectively.

b) **Identify target user' s nearest neighbors within the user cluster.** We generate enhanced similarity between the target user and other users. Within user cluster U_a, we select the top k users with the highest correlation coefficients to target user a to form the target user' s nearest neighbor set $\hat{N}(a)$, as shown in Equation (9):

$$\hat{N}(a) = \{b_j | b_j \in \text{Desc}(\hat{\text{Sim}}(a,b), b \in U_a), j \in [1,k]\}$$

where b_j represents the j-th user in descending order of correlation coefficient with user a, and $\text{Desc}(\hat{\text{Sim}}(a,b), b \in U_a)$ represents the sequence of users in U_a sorted by similarity to user a in descending order.

c) **Generate item deviation score using weighted average deviation.** After obtaining the target user' s nearest neighbors, we use weighted average deviation as the item deviation score ID_{an} for user a in user cluster U_a on target item I_n, as shown in Equation (10):

$$\text{ID}_{an} = \frac{\sum_{b \in \hat{N}(a)} \hat{\text{Sim}}(a,b) \cdot (r_{bn} - \bar{r}_b^{C_n})}{\sum_{b \in \hat{N}(a)} \hat{\text{Sim}}(a,b)}$$

where r_{bn} represents user b' s rating for item I_n in user cluster U_a, and ID_{an} represents the item deviation score of user a in user cluster U_a for target item I_n.

2.3 Calculating User Preference Score

The item deviation score is generated based on similarity, but its effectiveness is limited by co-rated items. When co-rated items are few or nonexistent, the item deviation score loses its 调节 ing significance. Therefore, we consider calculating the user preference score, which reflects users' preference degrees for different items. The algorithm first derives the user type preference score to 挖掘 user type preferences, then calculates the topic preference score, and finally computes the user preference score. The calculation process is as follows:

a) **Generate user type preferences from user type preference scores.** User type preferences are based on user type preference scores, which represent the contribution proportion of different types to the total rating sum—i.e., the proportion of influence each type has on the total score, as shown in Equation (11):

$$p_{ai} = \frac{\sum_{j=1}^{m} s_{aj} \cdot r_{aj}}{\sum_{i=1}^{t} \sum_{j=1}^{m} s_{aj} \cdot r_{aj}}$$

where p_{ai} represents user a' s preference score for type i, s_{aj} represents the number of comments user a makes on type j, r_{aj} represents user a' s rating for item j, t represents the number of types, and m represents the number of items. s_{aj} is each entry in the user-type rating count matrix $S_{a\times m}$, obtained by multiplying corresponding entries from the user-item membership matrix $R_{a\times n}$ and the item-type membership matrix $I_{n\times m}$. In $R_{a\times n}$, $r_{aj} = 1$ when user a has rated item j, otherwise $r_{aj} = 0$. Similarly, in $I_{n\times m}$, $i_{jm} = 1$ when item j belongs to type m, otherwise $i_{jm} = 0$, as shown in Equation (12):

$$s_{aj} = \sum_{j=1}^{n} r_{aj} \cdot i_{jm}$$

User type preferences include likes and dislikes. Z-score standardization reflects the degree of difference between a value and the mean, and its positive/negative results can precisely indicate user preferences for different types. Therefore, we apply Z-score normalization to users' type preference scores, as shown in Equation (13):

$$\hat{p}_{ai} = \frac{p_{ai} - \mu_a}{\sigma_a}$$

where $\hat{p}_{ai}$ represents user a' s preference for type i, μ_a represents the mean of user a' s type preference scores, and σ_a represents the standard deviation of user a' s type preference scores.

b) **Sequentially calculate topic preference score and user preference score.** We first compute the user' s topic preference score, which reflects

the user' s preference degree for topics. It is obtained by multiplying corresponding entries from the user-type preference matrix $\hat{P}_{a\times m}$ and the type-topic probability matrix $D_{m\times t}$ to generate the user-topic preference score, then summing the user-topic preference scores within the same item topic cluster CN_n, and finally using averaging to generate the user preference score UP_{an}, as shown in Equation (14):

$$UP_{an} = \frac{1}{|CN_n|} \sum_{t\in CN_n} \sum_{i=1}^{m} \hat{p}_{ai} \cdot d_{it}$$

where d_{it} represents the probability of type i corresponding to topic t in $D_{m\times t}$, and $|CN_n|$ represents the number of topics in item topic cluster CN_n.

2.4 Generating Final Predicted Rating

The predicted rating for a target item is generated based on the mean of existing ratings in the item cluster where the target item resides, with the addition of item deviation score and user preference score. The weight coefficient λ 调节 s the two prediction components to obtain the final predicted rating $\hat{r}_{an}$ for user a on item I_n, as shown in Equation (15):

$$\hat{r}_{an} = \bar{r}_{C_n} + \lambda \cdot \mathrm{ID}_{an} + (1-\lambda) \cdot UP_{an}$$

where $\bar{r}_{C_n}$ represents the mean of existing ratings in item cluster C_n for user a' s predicted item. The final predicted rating is influenced by the weight coefficient λ. When $\lambda = 0$, the predicted rating is only regulated by the user preference score; when $\lambda = 1$, it is only regulated by the item deviation score. The specific value of λ needs to be determined experimentally.

IUCF Algorithm Description

Algorithm: Item Deviation and User Preference Combination Filtering (IUCF)
Input: User-item rating matrix $R_{a\times n}$, item set I, user set U, weight coefficient λ
Output: Predicted rating $\hat{r}_{an}$ for target item

a) Perform LDA topic modeling on I to obtain item-type topic distribution $\hat{Z}_{t\times m}$ and topic-type probability matrix $\hat{Z}_{n\times m}$. Generate item clusters C_n from $\hat{Z}_{n\times m}$ and topic item clusters CT_t.

b) Digitally encode each user U_i' s basic information in U, perform K-means clustering on the encoded data, and generate user clusters U_a.

c) Within C_n, calculate enhanced rating similarity between users using Pearson correlation coefficient based on $R_{a\times n}$. Within U_a, generate target user

a's nearest neighbors $\hat{N}(a)$ based on $\hat{\mathrm{Sim}}(a,b)$ in descending order. Finally, generate item deviation score ID_{an} using weighted average deviation.

d) Multiply user-item membership matrix $R_{a\times n}$ and item-type membership matrix $I_{n\times m}$ to obtain user-type rating count matrix $S_{a\times m}$. Generate user type preference score p_{ai}, then apply Z-score standardization to obtain user type preference $\hat{p}_{ai}$.

e) Establish user-type preference matrix $\hat{P}_{a\times m}$ from $\hat{p}_{ai}$. Multiply $\hat{P}_{a\times m}$ with the transpose of type-topic probability matrix $D_{m\times t}$ to generate user-topic preference score. Sum user-topic preference scores within the same item topic cluster CN_n and use averaging to generate user preference score UP_{an}.

f) Use the mean $\bar{r}_{C_n}$ of existing ratings in target user a' s belonging cluster C_n as the baseline, and generate the final predicted rating $\hat{r}_{an}$ for item I_n by 调节 ing with weight coefficient λ.

Time Complexity Analysis of IUCF

Let n be the number of items, m the number of item types, t the number of topics, g the number of Gibbs iterations, a the number of users, q the number of user basic information attributes, k the number of K-means clusters, d the number of K-means iterations, and k' the number of nearest neighbors. With fixed numbers of users and items, the algorithm consists of offline and online components.

Offline Component:
Step a) LDA topic modeling to generate item clusters has time complexity $O(gnmt)$.
Step b) K-means clustering to generate user clusters has time complexity $O(daqk)$.
Step c) Calculating enhanced rating similarity within item clusters has time complexity $O(an^2)$.
Step d) Generating user type preferences from user type preference scores has time complexity $O(anm)$.

Online Component:
Step c) Identifying target user' s nearest neighbors within user clusters and generating item deviation score using weighted average deviation has time complexity $O(ak' + ak')$.
Step d) Generating user preference score from topic preference scores has time complexity $O(amt)$.
Step e) Generating final rating has time complexity $O(1)$.

In summary, the offline component contains two independent branches (ac and bd). Since the time complexity of LDA topic modeling is much greater than that of K-means, the overall offline complexity is $O(gnmt + an^2 + anm)$. The online complexity is $O(ak' + amt)$.

3 Experiments

3.1 Experimental Setup

The experimental dataset adopts the MovieLens 100K dataset developed and maintained by the GroupLens research group at the University of Minnesota, containing over 100,000 ratings from 943 users on 1,682 movies, along with movie genres and user attributes. We use 80% of the dataset as the training set and the remaining 20% as the test set, conducting experiments with five-fold cross-validation.

The experimental environment consists of an Intel Core i5 processor, 8GB RAM, Windows 7 x64 operating system, with algorithms implemented in Python 3.5.

3.2 Experimental Metrics

Accuracy is measured using Mean Absolute Error (MAE), which directly reflects recommendation quality. A smaller MAE indicates better recommendation accuracy, as shown in Equation (16):

$$\mathrm{MAE} = \frac{1}{|N|} \sum_{a,n \in N} |\hat{r}_{an} - r_{an}|$$

where $\hat{r}_{an}$ represents the predicted rating for user a on item n, r_{an} represents the actual rating, and N represents the number of predicted items.

3.3 Parameter Determination Experiments

The algorithm requires determining three parameters: K-means cluster count, LDA topic number, and weight coefficient λ.

3.3.1 Determining K-means Cluster Count To determine the appropriate K-means cluster count, we employ the elbow method, testing cluster counts of 2, 3, 4, 5, 6, 7, and 8, and calculating the corresponding SSE values. The results are shown in Figure 2 [Figure 2: see original paper].

In Figure 2, as the cluster count increases, the SSE value decreases sequentially. The plot shows that when the cluster count increases from 2 to 3, the SSE value drops by approximately 300, while the decrease rate slows significantly after 3, with the point at 3 being the inflection point of the elbow curve. Therefore, a cluster count of 3 yields the best clustering effect.

3.3.2 Determining LDA Topic Number To determine the appropriate LDA topic number, we set $\lambda = 1$ (i.e., using only the item deviation component) to identify the optimal topic number. Fixing nearest neighbor counts at 10, 30,

and 50, and using Gibbs sampling, we test topic numbers of 12, 13, 14, 15, 16, 17, and 18. Following the practice in 文献 [16], we set $\alpha = 50/T$ and $\beta = 0.01$. The results are shown in Figure 3 [Figure 3: see original paper].

Figure 3 shows that with nearest neighbor counts of 10, 30, and 50, the MAE is minimized when the topic number is 15, indicating that setting the topic number to 15 is most beneficial for ensuring algorithm performance regardless of neighbor count. Therefore, the optimal topic number for the algorithm is 15.

3.3.3 Determining Weight Coefficient λ To determine the optimal weight coefficient λ, we set nearest neighbor counts to 10, 30, and 50, and calculate the MAE of the IUCF algorithm under different λ values by 调节 ing λ. The results are shown in Figure 4 [Figure 4: see original paper].

Figure 4 demonstrates that for nearest neighbor counts of 10, 30, and 50, the IUCF algorithm achieves minimum MAE when $\lambda = 0.6$, indicating optimal recommendation quality. Therefore, the weight coefficient λ is set to 0.6 in experiments.

3.4 Algorithm Comparison Experiments

To comprehensively compare the recommendation accuracy of IUCF with other algorithms, we select four algorithms for comparison: traditional User-based Collaborative Filtering (UCF), Item-based Collaborative Filtering (ICF), the Feature Transfer and Probabilistic Matrix Factorization recommendation algorithm (FTMF) proposed in 文献 [4], and the Enhanced Collaborative Filtering Adopting Trust Network (ECFATN) proposed in 文献 [11]. The experimental results are shown in Figure 5 [Figure 5: see original paper].

In Figure 5, compared with traditional UCF and ICF, IUCF shows significantly lower MAE, demonstrating that the proposed algorithm indeed improves the accuracy of traditional collaborative filtering algorithms. Compared with ECFATN, IUCF achieves lower MAE under different neighbor counts, indicating superior performance. However, compared with FTMF, when the neighbor count exceeds 30, IUCF' s MAE becomes higher than FTMF' s, possibly because too many neighbors in the item deviation component 反而 reduce algorithm accuracy. Nevertheless, when the neighbor count is less than 30, IUCF' s MAE is lower than FTMF' s, showing that the proposed algorithm improves accuracy with smaller neighbor counts.

To verify the time efficiency difference between IUCF and other algorithms, we compare it with UCF and ICF by generating recommendations for 20, 40, and 60 users. The runtime comparison is shown in Table 1 .

Table 1 shows that IUCF is significantly more time-efficient than traditional item-based collaborative filtering, but has longer runtime than user-based collaborative filtering. This is because similarity calculations for item-based collaborative filtering must be performed online, while those for user-based collaborative

filtering can be completed offline.

In summary, the IUCF algorithm achieves improvements in both recommendation accuracy and time efficiency.

---

4 Conclusion

Accurately measuring item deviation and user preference significantly impacts recommendation quality. By limiting the computation scope and excluding irrelevant items and users, the proposed algorithm effectively alleviates the accuracy issues caused by mutual influences among correlation factors between items and users in collaborative filtering recommendation, achieving satisfactory results.

Future work may employ natural language processing techniques to extract sentiment words from user reviews, thereby 挖掘 ing item deviation and user preferences to further improve recommendation system accuracy and make recommendation results more persuasive.

References

[1] Chen Wei, Hsu W, Lee M L. A unified framework for recommendations based on quaternary semantic analysis [C]// Proc of the 34th international ACM SIGIR conference on Research and development in Information Retrieval. New York: ACM Press, 2011: 1023-1032.

[2] Shi Min, Liu Jianxun, Zhou Dong, et al. A probabilistic topic model for mashup tag recommendation [C]// Proc of IEEE International Conference on Web Services. 2016: 444-451.

[3] Qiao Zhi, Zhang Peng, He Jing, et al. Combining geographical information of users and content of items for accurate rating prediction [C]// Proc of the 23rd International Conference on World Wide Web. New York: ACM Press, 2014: 361-362.

[4] Zhao Zhilin, Wang Changdong, Wan Yuanyu, et al. FTMF: recommendation in social network with feature transfer and probabilistic matrix factorization [C]// Proc of International Joint Conference on Neural Networks. 2016: 847-854.

[5] Zhou Xiuze, Wu Shunxiang. Rating LDA model for collaborative filtering [J]. Knowledge-Based Systems, 2016, 110: 135-143.

[6] Yuan Fuyong, Wen Zhihui, Liang Shunpan, et al. Integrating Item Category Into Weighted Slope One Algorithm [J]. Journal of Chinese Mini-Micro Computer Systems, 2017, 38 (09): 2090-2095.

[7] Liu Huiting, Chen Yan, Xiao Huihui. Matrix factorization recommendation algorithm based on users'preference [J]. Journal of Computer Applications, 2015, 35 (S2): 118-121.

[8] Herlocker J L, Konstan J A, Borchers A, et al. An algorithmic framework for performing collaborative filtering [C]// Proc of International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 1999: 230-237.

[9] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet allocation [J]. Journal of Machine Learning Research, 2003, 3 (3): 993-1022.

[10] Casella G, George E I. Explaining the Gibbs sampler [J]. The American Statistician, 1992, 46 (3): 167-174.

[11] Li Yichen, Chen Li, Shi Chenchen, et al. Enhanced collaborative filtering adopting trust network [J]. Application Research of Computers, 2018, 35 (01): 116-120.

[12] He Ming, Sun Wang, Xiao Run, et al. Collaborative filtering recommendation algorithm combining clustering and user preferences [J]. Computer Science, 2017, 44 (S2): 391-396.

[13] Luo Chun. Re-partitioning population age group and its implications [J]. Population Research, 2017, 41 (05): 16-25.

[14] Wood J C, Wood M C. Joseph M. Juran: Critical evaluations in business and management [M]. [S. l.]: Psychology Press, 2005.

[15] Bholowalia P, Kumar A. EBK-means: A clustering technique based on elbow method and k-means in WSN [J]. International Journal of Computer Applications, 2014, 105 (9).

[16] Griffiths T L, Steyvers M. Finding scientific topics [J]. Proceedings of the National academy of Sciences, 2004, 101 (suppl 1): 5228-5235.

Note: Figure translations are in progress. See original paper for figures.