Computing and Informatics, Vol. 39, 2020, 141–155, doi: 10.31577/cai_2020_1-2_141

TIME-SENSITIVE COLLABORATIVE FILTERING ALGORITHM WITH FEATURE STABILITY

Shanchen Pang, Shihang Yu, Guiling Li
Sibo Qiao, Min Wang

Shandong University of Science and Technology
No.579 Qianwangang Road
Qingdao, China
e-mail: qd-liguiling@163.com

Abstract. In the recommendation system, the collaborative filtering algorithm is widely used. However, there are lots of problems which need to be solved in recommendation field, such as low precision, the long tail of items. In this paper, we design an algorithm called FSTS for solving the low precision and the long tail. We adopt stability variables and time-sensitive factors to solve the problem of user's interest drift, and improve the accuracy of prediction. Experiments show that, compared with Item-CF, the precision, the recall, the coverage and the popularity have been significantly improved by FSTS algorithm. At the same time, it can mine long tail items and alleviate the phenomenon of the long tail.

Keywords: Collaborative filtering, recommendation algorithm, long tail, time-sensitive

1 INTRODUCTION

With the upsurge of the 5G era, edge computing has also exploded. Edge computing complements cloud computing, providing better real-time, faster data processing capabilities, lower processing costs, lower network bandwidth requirements, and better privacy protection for mobile edge devices [1, 2, 3]. Edge computing brings the functions of the cloud computing center closer to the user-side network, shortens the space distance for users to obtain services, and greatly reduces the delay with services [4]. Nowadays, there are many new situations on the Internet.

Mobile e-commerce is becoming more and more popular. Privacy protection is getting more and more attention in the public. Netizens are drowned in the flood of information and cannot get the information they interest themselves in. However, the recommendation system can recommend the information data according to the history of the user, actively recommend the information that may be of interest to the user, and discover the information valuable to the user in the information flow. The recommendation algorithm is deployed on the edge computing server, which can provide users with real-time recommendations, and can meet the needs of users to protect privacy, and can extract the information of the real-time needs of users. The accuracy of the recommendation system depends mainly on the performance of the recommendation algorithm.

Recommendation algorithms are mainly divided into the neighborhood-based model [5], the latent factor model [6] and the graph-based model [7, 8]. The neighborhood-based model is most widely used in e-commerce scene. This paper mainly aims to improve the neighborhood-based model. The neighborhood-based model is the simplest and easiest algorithm among recommendation algorithms, which is researched deeply in academia and applied most widely in industry. Neighborhood-based algorithms can be divided into two categories: user-based collaborative filtering (User-CF) [9, 10] and item-based collaborative filtering (Item-CF) [11]. The recommendation process of user-based collaborative filtering algorithm is that when user A needs to get some recommendations, firstly, this algorithm will seek out user group B whose interest is like user A's, and then recommend items to A that user group B likes but A has not bought. This method is mainly applicable to areas where the number of users is small, the timeliness is not strong, and the user's personalized interest is not conspicuous. While the recommendation process of item-based collaborative filtering algorithm is that when user A purchases item a, firstly, the algorithm will find item set b which is like item a, and recommend items in item set b to user A which has not been purchased by user A. This method is suitable for the areas where the number of items is obviously less than the number of users, long tail items are abundant, and users have strong personalized needs. Item-based collaborative filtering algorithm is more widely applied in industry circles than user-based collaborative filtering algorithm. The algorithm in this paper is an improvement of the item-based collaborative filtering algorithm.

2 RELATED WORK

Lots of scholars have done numerous studies on collaborative filtering based on items. Huang and other scholars combined the user theme model with the item theme model, and considered the tag information of the item and the users' behavior data [12]. They proposed a hybrid recommendation algorithm based on the item collaborative filtering model, which improved the diversity and accuracy of the recommendation. Nikolakopoulos et al. proposed a new random migration method, which overcomes the obstacle of fast gait convergence to the graph center by using the spectral characteristics of uncoupled Markov chains [13]. With this method, they prolonged the follow-up effect of users' previous preferences on walking, allowed pedestrians to explore more basic networks, and improved the recommendation quality. Chen et al. put forward the Attentive Collaborative Filtering (ACF) model, which used groupware-level attention module to extract network from content features, and used project-level attention module to mark preferences of projects [14]. The experimental results show that ACF is superior to CF. Wei et al. used deep neural network to collect item content features, and considered the temporal dynamics of users' preferences and temporal dynamics of item features [15]. They used collaborative filtering to build a model, which improved the prediction accuracy of recommendation system. Dong et al. put forward a hybrid model through learning the efficiency of feature extraction, which can effectively know the users' depth and the potential features of the project from the score matrix [16]. Li et al. combined the decomposition of probability matrix with the stackable automatic which has the feature of edge noise reduction [17]. With this method, the problem of insufficient feature extraction caused by data sparsity has been solved. Nilashi et al. came up with an incremental updating multi-criteria collaborative filtering method based on clustering and regression [18]. This method automatically detects and subdivides users by clustering, the learning preference model is subdivided for each user, and the preference model can be updated incrementally. Although the methods which are talked about above got good forecasting results, the model is too complex, and it is difficult to calculate the similarity of items. At the same time, the long tail [19] distribution of items has not been taken into consideration.

The FSTS algorithm proposed in this paper not only extracts the features of items, but also considers the stability of the features of items. Meanwhile, a time-sensitive factor is added to make the algorithm time-sensitive to combat the phenomenon of interest drift [20]. This algorithm can effectively improve the long tail pheonmenon of items while ensuring the prediction accuracy.

3 FSTS ALGORITHM MODEL

FSTS algorithm constructs feature vectors of items by analyzing the frequency of items purchased by users and their rating. At the same time, because the user's interest will change with the passing of time, which is called interest drift, this algorithm adds time influence factor to solve user's interest drift problem. The main steps of FSTS include data preprocessing, feature extraction of items, time-sensitive factor antagonism, rule base generation and item recommendation. The main process is shown in Figure 1.

1. Data preprocessing stage: This stage mainly handles invalid ratings and invalid users. There are some invalid ratings in the user-item scoring matrix, such as those higher than the maximum value and those with error codes caused by some factors, which will affect the final rule base and lead to the inaccuracy of the recommendation prediction. Invalid users refer to those who do not give

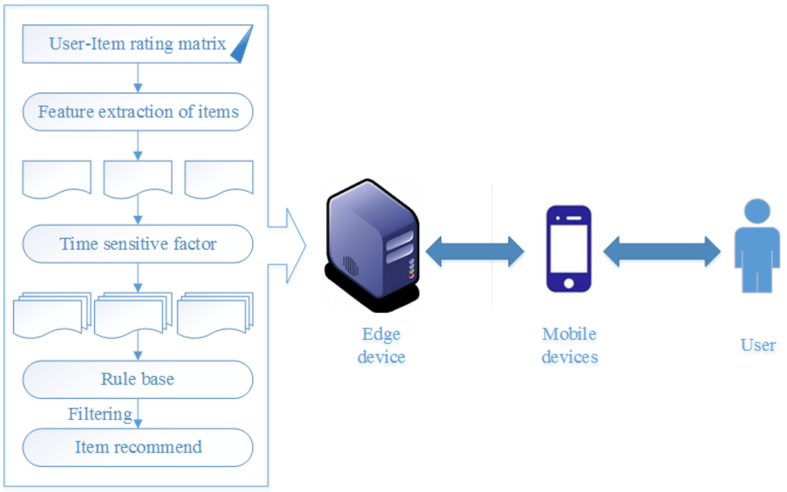

Figure 1. FSTS algorithm model

any marks on items or who give marks on all items. These users who are called zombie users are likely to attack the recommendation system. These users are meaningless to the final prediction, and even will affect the final prediction results, so they should be eliminated.

2. Feature extraction of items: The frequency of items being purchased and the ratings of items are regarded as the feature of items. The feature vectors of each item are calculated by using the user-item rating matrix. Then the feature vectors of all items are obtained, and the preliminary rule base is gained.
3. Adding time impact factors to confront: User's interest is changed with the passing of time, which is interest drift phenomenon. This algorithm adds time impact factor to solve interest drift problem, not only excavating users' long-term interest, but also discovering users' short-term interest.
4. Building the rule base. The algorithm is used to calculate the similarity model of items and build the recommendation rule base. When recommending items to users, the corresponding rules can be picked up directly from the rule base.
5. Item recommendation: When users browse, collect, add items into shopping carts and purchase items, some invalid rules are filtered out according to the rules related to the current items in the rule base. For example, Top-N, which is highly relevant to the current items but has been purchased by users, is recommended to users.

In the paper, we make the following assumptions:

1. In this recommendation system, we do not consider the cold start of systems.
2. In the paper, we do not consider the cold items and the cold users.

3. In the paper, we consider the single recommendation scene. And we do not consider the multiple recommendation scenes.

4 FSTS RECOMMENDATION ALGORITHMS

4.1 Problem Modeling

Suppose that there are a certain number of users and a certain number of items that were purchased by users. The set of these users is expressed as $U = \{u_1, u_2, \dots, u_N\}$ and the set of items is expressed as $S = \{s_1, s_2, \dots, s_M\}$. The user u_i scored the item s_i as a_{ij}, so the user's corresponding interest vector is expressed as $v_i = (a_{i1}, a_{i2}, \dots, a_{ij})$. User-item rating matrix m whose size is $N \times M$, is composed of all users' interest vectors. For users u_i and items s_j with purchasing relationship, the corresponding rating is $a_{ij} > 0$, while for users and items without purchasing relationship, the corresponding rating is $a_{ij} = 0$.

4.2 The Description of the Algorithm

4.2.1 Features of Items

For recommendation system, if a user has not purchased any items or the user has purchased all items, it not only increases the difficulty for calculation, but also reduces the precision of prediction, so the existence of this user is meaningless. Then for a normal user, the more items he has purchased, the less influence he will have on the generation rules of associated rule base. Therefore, user's activity is defined as A, which is shown in Formula (1):

$$A_i = \log_2 (1 + n_i) . \tag{1}$$

In this formula, the activity of the user i is expressed as A_i. n_i represents the total number of items purchased by the user i.

In this paper, Formula (1) is used to perform feature extraction on users to distinguish different users, and thus to distinguish the different users and the different contributions of different users to the similarity of items.

Different users' rating criteria are also different. For example, if the full mark is ten points, some users will think that 9.5 is high enough. These users may not give 10 points for their favorite items, or give very little full marks, while other users could easily give 10 points for items they were satisfied with. Thus, it can be known that the rating is a process which is mixed with subjective factors. Users are satisfied with the same item, but the marks cannot objectively reflect the user's satisfaction. Therefore, this algorithm is designed to standardize the user's rating in order to eliminate the user's own bias on the item.

Directly related to the item's features is the user's rating to the item and the activity of the user who purchased the item. Therefore, the feature matrix of items is defined as $F = (F_1, F_2, \ldots, F_M)^T$, the feature vector of the item j is defined as $F_j = (f_{j1}, f_{j2}, \ldots, f_{jN})$ and the feature value of the item j at the user i is f_{ji}, which is shown in Formula (2):

$$f_{ji} = \alpha \times \frac{R_{ji} - R_{imin}}{R_{imax} - R_{imin} + 1} + \beta \times \frac{A_i - A_{jmin}}{A_{jmax} - A_{jmin} + 1}. \tag{2}$$

In this formula, R represents the user's rating on the item, R_{ji} represents the user i's rating on the item j, R_{imin} means the minimum value of all the ratings given by the user i, and R_{imax} means the maximum value of all the ratings made by the user i, A represents the activity of users, A_i is the activity of user i, A_{jmin} represents the minimum activity of all users who have purchased item j, and A_{jmax} is the maximum activity of all users who have purchased item j. α, β are the formula parameters, and meet the formula requirements: $0 \le \alpha \le 1$, $0 \le \beta \le 1$, $\alpha + \beta = 1$. In this experiments, $\alpha = 0.5$ and $\beta = 0.5$.

Different users have different ratings on the same item. Some items have been given higher ratings, some items have been given lower ratings, and some items have more differences in ratings. In this paper, the variance of ratings is used to describe the change of item' rating, that is, the stability of item rating. The lower the stability of the item rating, the more consistent the rating of all users is, which means the item is less significant for calculating its contribution to the similarity with other items. The higher the stability of item rating, the more inconsistent the rating of all users is. That is, the more inconsistent the user likes or not, which means the item is more useful for calculating its contribution to the similarity with other items. The description of the rating stability of item j is shown in Formula (3):

$$F\_S(j) = \frac{\sum_{i=1}^{N} (a_{ij} - \overline{a_{-j}})}{N}. \tag{3}$$

In this formula, $F\_S(j)$ represents the rating stability of the item j. a_{ij} represents the user i's rating on the item j and $\overline{a_{-j}}$ is the average value of all users' rating on the item j. N represents the total number of users.

4.2.2 Time-Sensitive Factors

As the time goes by, the user's interest is likely to change, that is called user's interest drift phenomenon. An item purchased by a user on the spot has a very low or even no correlation with the item he purchased long ago. While recommendation is timeliness, which means when the user decides to choose or purchase the current item, items related to or similar to the current item should be recommended immediately. Therefore, time-sensitive factor $e^{-\delta|t_{ij}-t_{ik}|}$ is added to this algorithm to solve interest drift caused by time passing. This factor is on a daily basis. t_{ij} represents the time when the user i purchases the item j, δ means the time-sensitive coefficient, which satisfies the requirement: $0 < \delta < 1$. In this expriments, $\delta = 1/7$. The closer the time for user i to purchase item j and item k, the greater the similarity between item j and item k.

4.2.3 FSTS Algorithm

The FSTS algorithm assigns linear weight to the stability of item rating and the time-sensitive factor, in order to influence the feature vector of the item, which is shown as Formula (4):

$$FSTS_{jk} = \frac{F_j \ldots F_k}{F_j F_k} \times \left(F\_S(j) \times F\_S(k) + \sum_{i \in U_j \cap U_k} e^{-\delta |t_{ij} - t_{ik}|} \right). \tag{4}$$

In this formula, $FSTS_{jk}$ represents the similarity between item j and item k. U_j represents a set of users who have all purchased item j and U_k is a set of users who have all purchased item k. This algorithm adds the feature stability of items and time-sensitive factor, which is called FSTS algorithm. The algorithm considers both the rating stability of items and time-sensitive factor, and it can flexibly adjust the influence weights of them for different application scenarios, which has achieved good prediction results.

5 EXPERIMENTAL ANALYSIS

5.1 Experimental Data Set

This experiment uses the public movie data set MovieLens provided by GroupLens, which is a specialized research laboratory of recommendation system. This experiment uses a 20 M MovieLens data set, which contains 138 000 users giving 20 000 000 marks to 27 000 movies. In this experiment, the data set is divided into test set and training set according to the ratio of 2 : 8.

5.2 Evaluation Criteria

There are four evaluation criteria in this experiment: Precision, Recall, Popularity and Coverage, which are used as performance indicators to evaluate the final recommendation results of the experiment. The four criteria are shown in Formulas (5), (6), (7), and (8):

$$\text{Precision} = \frac{\sum_U |R(u) \cap T(u)|}{\sum_U |R(u)|}, \tag{5}$$

$$\text{Recall} = \frac{\sum_U |R(u) \cap T(u)|}{\sum_U |T(u)|}, \tag{6}$$

$$\text{Popularity} = \sum_M \log_2 (1 + \text{sum}(m)), \tag{7}$$

$$\text{Coverage} = \frac{|U_{u \in U} R(u)|}{|I|}. \tag{8}$$

In these formulas, $R(u)$ represents the item collection recommended to users, $T(u)$ means the collection of items that users like in the test set; U represents the set of all users in the test set, I means the number of all users in the data set, M represents a collection of movies in the recommendation list, and $sum(m)$ is the times of the movie m appearing in the training set.

Precision and recall describe the prediction accuracy of recommendation results for users. The higher are the precision and recall, the better are the recommendation results. Popularity and coverage describe the composition of the items of the recommendation results. The lower is the popularity and the higher is the coverage, the better are the recommendation results.

5.3 Result Analysis

In this paper, FSTS is compared with Item-CF algorithm. The numbers of recommendation neighbors are respectively set to 5, 10, 15, 20, 25, 30, 35 and 40. The experimental results show that FSTS algorithm outperforms Item-CF algorithm in all aspects of prediction performance. Figure 2 shows that compared with Item-CF algorithm, the precision of FSTS algorithm is significantly better than that of Item-CF algorithm. At that time, the prediction accuracy of FSTS algorithm increased by 7.6 % when $K = 5$. As the value of K increases, the precision of the FSTS algorithm decreases as the similarity of the recommended items decreases. Figure 3 shows that the recall of FSTS algorithm is also better than that of Item-CF algorithm. At that time, the recall of FSTS algorithm increased by 6.1 % when $K = 25$. As the value of K increases, the similarity of the recommended items decreases, so the recall of FSTS algorithm increases which means the precision decreases.

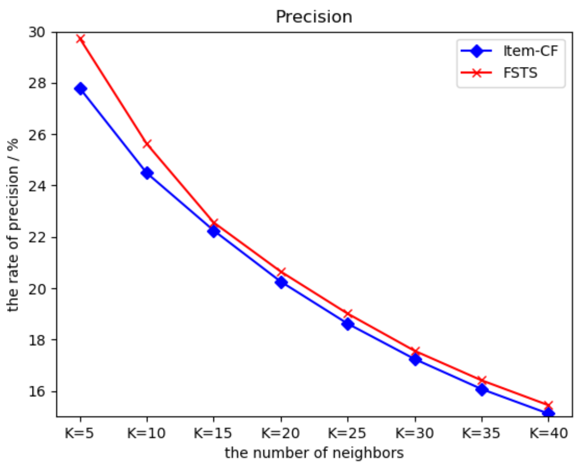

Figure 2. Comparison of precision between FSTS and item-CF

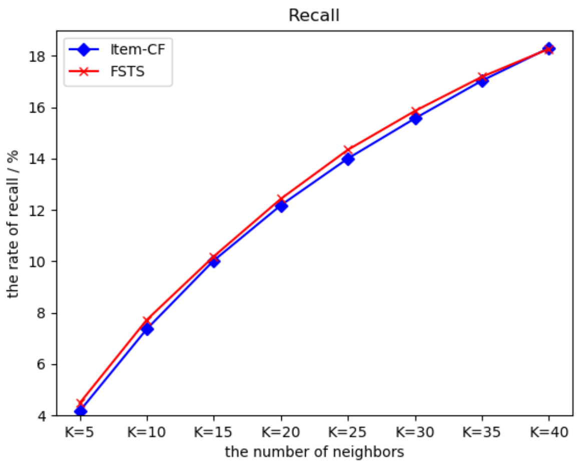

Figure 3. Comparison of recall between FSTS and item-CF

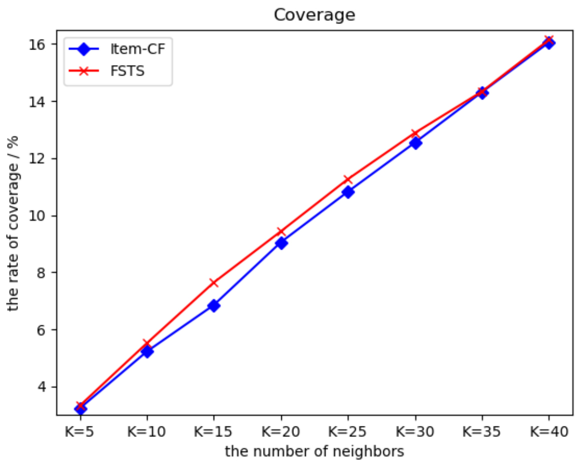

Figure 4. Comparison of coverage between FSTS and item-CF

Coverage and popularity reflect the overall quantity of items. Figure 4 shows that compared with Item-CF algorithm, the coverage of FSTS algorithm has been improved. When $K = 15$, the coverage of FSTS algorithm has increased by 15.1 %. As the value of K increases, the number of items recomended to users increases, so the coverage increases. Figure 5 shows that the proportion of popular items recommended by the FSTS algorithm is significantly lower than that recommended by the

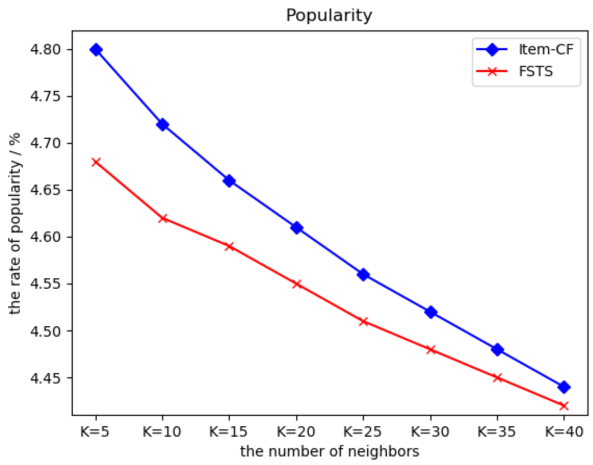

Figure 5. Comparison of popularity between FSTS and item-CF

Item-CF algorithm. At the same time, the popularity of items in the recommendation list of the FSTS algorithm decreased by 4.3 %, when $K = 5$. As the value of K increases, the popularity of items decreases, so the popularity of the FSTS algorithm decreases.

Long tail of items is also alleviated by this algorithm recommendation. Figure 6 shows the distribution of the movies watched, and Figure 7 shows the long

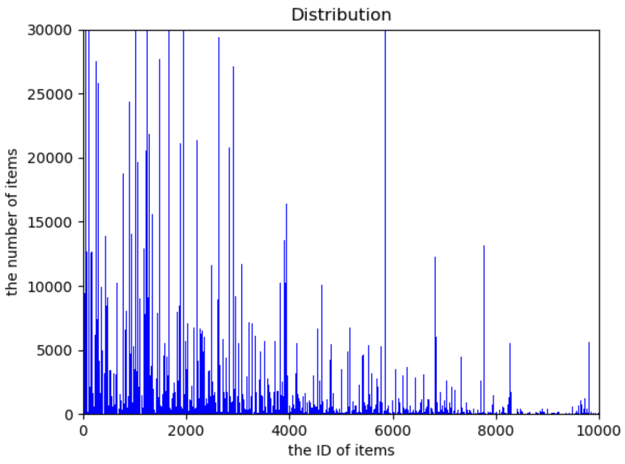

Figure 6. The distribution of item

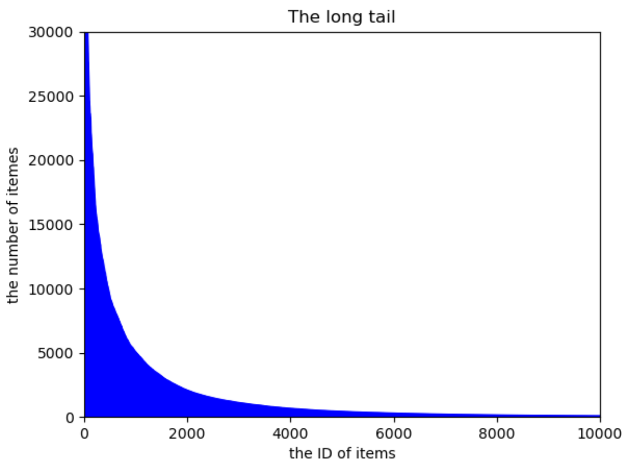

Figure 7. The long tail

tail of the items before the algorithm is used. It can be known that the data set is consistent with the long tail, and the long tail is obvious. Figure 8 shows that the long tail distribution of items has been significantly alleviated after multiple recommendation using FSTS algorithm, that is, items at the tail place have been more fully recommended and potential interests of users have been fully excavated.

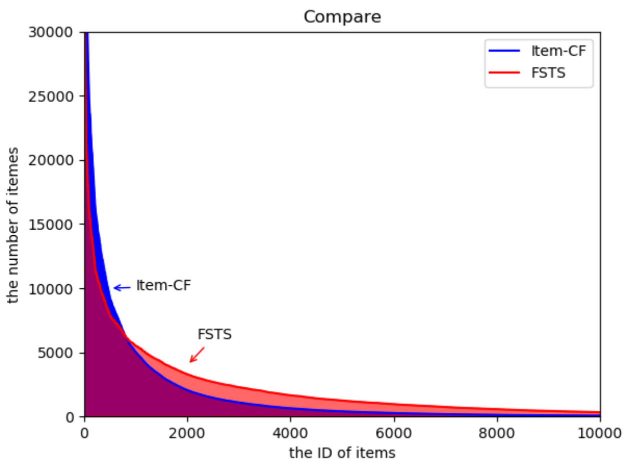

Figure 8. Comparison of long tail distribution between item-CF and FSTS

6 SUMMARY AND OUTLOOK

This paper is aimed at the characteristics of the edge devices and FSTS algorithm is designed in this paper. The algorithm takes into consideration both the stability of item features and users' interest drift. In the prediction process, the feature vector and time-sensitive factor are added to deeply extract items features and deal with the drift of user interest. Experiments show that FSTS algorithm improves both prediction performance and algorithm performance with low time complexity.

However, the dynamic transformation of the time-sensitive factor is weak. Next, we will strengthen the dynamic transformation of this factor, and deploy the algorithm to the recommendation system, and further optimize the algorithm through the feedback mechanism of the system, in order to obtain more accurate prediction data and higher efficiency. And in this paper, we do not consider the cold start scenes. So the FSTS algorithm does not fit the cold start scenes. Next, we will do the work about the cold start of system, the cold users and the cold items. Beside, we consider to use CNN net to accelerate the algorithm in the next step.

REFERENCES

[1] Pang, S.—Qiao, S.—Song, T.—Zhao, J.—Zheng, P.: An Improved Convolutional Network Architecture Based on Residual Modeling for Person Re-Identification in Edge Computing. IEEE Access, Vol. 7, 2019, pp. 106749–106760, doi: 10.1109/ACCESS.2019.2933364.

[2] Song, T.—Pang, S.—Hao, S.—Rodríguez-Patón, A.—Zheng, P.: A Parallel Image Skeletonizing Method Using Spiking Neural P Systems with Weights. Neural Processing Letters, Vol. 50, 2019, pp. 1485–1502, doi: 10.1007/s11063-018-9947-9.

[3] Pang, S.—Wang, M.—Qiao, S.—Wang, X.—Chen, H.: Fault Diagnosis for Service Composition by Spiking Neural P Systems with Colored Spikes. Chinese Journal of Electronics, Vol. 28, 2019, No. 5, pp. 1033–1040, doi: 10.1049/cje.2019.06.023.

[4] Pang, S.—Gao, Q.—Liu, T.—He, H.—Xu, G.—Liang, K.: A Behavior Based Trustworthy Service Composition Discovery Approach in Cloud Environment. IEEE Access, Vol. 7, 2019, pp. 56492–56503, doi: 10.1109/ACCESS.2019.2913432.

[5] Saelens, B. E.—Sallis, J. F.—Black, J. B.—Chen, D.: Neighborhood-Based Differences in Physical Activity: An Environment Scale Evaluation. American Journal of Public Health, Vol. 93, 2003, No. 9, pp. 1552–1558, doi: 10.2105/ajph.93.9.1552.

[6] Zhang, W.—Wang, J.—Feng, W.: Combining Latent Factor Model with Location Features for Event-Based Group Recommendation. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13), ACM, 2013, pp. 910–918, doi: 10.1145/2487575.2487646.

[7] Pang, S.—Chen, H.—Liu, H.—Yao, J.—Wang, M.: A Deadlock Resolution Strategy Based on Spiking Neural P Systems. Journal of Ambient Intelligence and Humanized Computing, 2019, 12 pp., doi: 10.1007/s12652-019-01223-3.

[8] Fouss, F.—Pirotte, A.—Renders, J.-M.—Saerens, M.: Random-Walk Computation of Similarities Between Nodes of a Graph with Application to Collaborative Recommendation. IEEE Transactions on Knowledge and Data Engineering, Vol. 19, 2007, No. 3, pp. 355–369, doi: 10.1109/TKDE.2007.46.

[9] Zhao, Z.-D.—Shang, M.-S.: User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. 2010 Third International Conference on Knowledge Discovery and Data Mining (WKDD '10), IEEE, 2010, pp. 478–481, doi: 10.1109/WKDD.2010.54.

[10] Wang, S.—He, S.—Yuan, F.—Zhu, X.: Tagging SNP-Set Selection with Maximum Information Based on Linkage Disequilibrium Structure in Genome-Wide Association Studies. Bioinformatics, Vol. 33, 2017, No. 14, pp. 2078–2081, doi: 10.1093/bioinformatics/btx151.

[11] Linden, G.—Smith, B.—York, J.: Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, Vol. 7, 2003, No. 1, pp. 76–80, doi: 10.1109/MIC.2003.1167344.

[12] Huang, L.—Lin, C.—He, J.—Liu, H.—Du, X.: Diversified Mobile App Recommendation Combining Topic Model and Collaborative Filtering. Journal of Software, Vol. 28, 2017, No. 3, pp. 708–720, doi: 10.13328/j.cnki.jos.005163 (in Chinese).

[13] Nikolakopoulos, A. N.—Karypis, G.: Boosting Item-Based Collaborative Filtering via Nearly Uncoupled Random Walks. arXiv preprint arXiv:1909.03579, 2019.

[14] Chen, J.—Zhang, H.—He, X.—Nie, L.—Liu, W.—Chua, T.-S.: Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17), ACM, 2017, pp. 335–344, doi: 10.1145/3077136.3080797.

[15] Wei, J.—He, J.—Chen, K.—Zhou, Y.—Tang, Z.: Collaborative Filtering and Deep Learning Based Recommendation System for Cold Start Items. Expert Systems with Applications, Vol. 69, 2017, pp. 29–39, doi: 10.1016/j.eswa.2016.09.040.

[16] Dong, X.—Yu, L.—Wu, Z.—Sun, Y.—Yuan, L.—Zhang, F.: A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems. Thirty-First AAAI Conference on Artificial Intelligence (AAAI '17), 2017, pp. 1309–1315.

[17] Li, S.—Kawale, J.—Fu, Y.: Deep Collaborative Filtering via Marginalized Denoising Auto-Encoder. Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM '15), ACM, 2015, pp. 811–820, doi: 10.1145/2806416.2806527.

[18] Nilashi, M.—Jannach, D.—bin Ibrahim, O.—Ithnin, N.: Clustering- and Regression-Based Multi-Criteria Collaborative Filtering with Incremental Updates. Information Sciences, Vol. 293, 2015, pp. 235–250, doi: 10.1016/j.ins.2014.09.012.

[19] Oestreicher-Singer, G.—Sundararajan, A.: Recommendation Networks and the Long Tail of Electronic Commerce. MIS Quarterly, Vol. 36, 2012, No. 1, pp. 65–83, doi: 10.2307/41410406.

[20] Yin, H.—Zhou, X.—Cui, B.—Wang, H.—Zheng, K.—Nguyen, Q. V. H.: Adapting to User Interest Drift for POI Recommendation. IEEE Transactions on

Knowledge and Data Engineering, Vol. 28, 2016, No. 10, pp. 2566–2581, doi: 10.1109/TKDE.2016.2580511.

Shanchen Pang received his graduation degree from the Tongji University of Computer Software and Theory, Shanghai, China, in 2008. He is Professor in the China University of Petroleum, Qingdao, China. His current research interests include theory and application of Petri net, service computing, trusted computing.

Shihang Yu received his graduation degree in the Shandong University of Science and Technology, Qingdao, in 2017. He is graduate student of the China University of Petroleum, Qingdao, China. His current research interests include recommendation system, data mining.

Guiling Li received her engineering Ph.D. degree in control theory and control engineering from the Shandong University of Science and Technology, Qingdao, China, in 2009 and 2014. In 2005, she began working in the Academy of Mathematics and Systems Science at Shandong University of Science and Technology, Qingdao, China. Her research interests are in the area of stochastic control.

Sibo Qiao received his B.Sc. degree from the Shandong University of Science and Technology, Qingdao, in 2017. He is graduate student of the China University of Petroleum, Qingdao, China. His current research interests include deep learning, person reidentification, and object detection.

Min Wang received her M.Sc. degree from the China University of Petroleum, Qingdao, China, in 2019. She is Ph.D. student of control science and engineering in the China University of Petroleum, Qingdao, China. Her current research interests include theory and application of Petri net, trusted computing.