2018 42nd IEEE International Conference on Computer Software & Applications

A software popularity recommendation method based on evaluation model

Yan Wang,Pei-xiang Bai,De-yu Yang,
Jian-tao Zhou*
Inner Mongolia Engineering Lab of Cloud Computing
and Service Software, College of Computer Science
Inner Mongolia University
Hohhot, China
cswy,pxb,deyu@imu.edu.cn,cszhoujiantao@qq.com

Xiaoyu Song
Department of ECE
Portland State University Portland
OR, USA
song@ece.pdx.edu

Abstract—**The software sharing platform in the Internet provides great convenience for the promotion, application and communication of software (especially source software). But there inevitably exists the problem of software quality on the open Internet platform. How the users choose software to download and use becomes a new challenge for software sharing platforms. Aimed at the above problems and challenges, the internal relations between the data collected on platform and experience of user are analyzed. And then a software popularity recommendation method based on evaluation model is presented. The method constructs two evaluation indexes based on the collected data on the platform, including attention-degree and satisfaction-degree; solves the problem of small sample data's influence on the accuracy of evaluation model by using the Wilson interval model and makes a tradeoff between the recommendation results of old and new software by using Newton cooling law. The experimental results show that the software popularity recommendation method based on evaluation model helps users to screen for software, which can effectively improve the service performance of software sharing platform.**

Keywords-software sharing platform; evaluation model; software popularity; recommendation

I. INTRODUCTION

In recent years, software as a carrier of information service in the information age has drawn much attention from users. According to statistics, up to the end of 2016, the number of newly published software exceeded ten million. With the rapid development of Internet technology, a large number of platforms or websites providing software downloading services are also emerging on the Internet. Explosive growth of software has provided great conveniences for users to use the software on the platform. Due to the openness of such platforms, people can not only download the software they need as a user but also upload the software sharing with other users developed by themselves as a publisher. It has a positively effect in the application and development of software to a large extent. Especially some open source code-sharing platforms for software developers and IT practitioners provide a tool for programmers to learn and communicate with each other, so that they can acquire and share knowledge or information in career development and establish career development circles, also can through software development to meet the rigid demand about technology commercialization.

However, due to the different quality of users in the platform, some users may upload resources and software with spam or malicious code, which makes such open platforms getting a hardest hit because of the proliferation and expansion of spam and virus Trojan. At the same, it seriously disrupts the normal network order, not only wasting platform resources, but also seriously affecting the user's participation experience. As a result, a huge loss for platform operators has brought. Therefore, how to select, install and operate the software in the downloading platform has become an important issue for users [1]. Meanwhile, the dramatic increase about the number of software has aroused people's concern about software evaluation and recommendation.

The traditional software sharing platform relies mainly on the manual analysis and comparison of the platform administrator to check the quality of the software. These software recommendation and rankings method are based on single data, such as software download or user feedback after software usage, not only without fully utilizing the collectable data which can influence the user experience in the platform, but also without considering the impact of these data on the user experience synthetically. So it is difficult to adapt to the automated evaluation and recommendation of software in the platform. For this reason, it have a great researching and practical significance to make full use of the data freely accessible in the software sharing platform which could affect the user experience of software to study the software evaluation model and the recommendation method. To resolve the problems and challenges mentioned above, this paper analyzes the inherent relationship between open source data and user experience in the platform, proposes a popularity recommendation method based on software evaluation in Internet platform. This method fully assesses the user's acceptance of the software according to the use of software, the evaluation of software and time factor in the platform.

The remainder of this paper is structured as follows. Section II introduces some mainstream algorithms for

* Corresponding author

978-1-5386-2666-5/18/$31.00 ©2018 IEEE
DOI 10.1109/COMPSAC.2018.00070

IEEE computer society

recommendation systems and points out a series of problems existing in these works. Section III elaborates our evaluation model based on degree of attention and degree of satisfaction. Section IV verifies the inherent relationship among various factors in the proposed model through experimental analysis based on simulation data. Section VI concludes this paper.

II. BACKGROUND AND RELATED WORKS

The recommendation system is the most relevant field to the issue studied in this paper. Its aim is to recommend the product or service which is most likely to be satisfied to a user based on the user's preference. It can be regarded as an intelligent decision supporting platform based on massive data analysis, and has been successfully used to deal with the problem of information overloading. A good recommendation method can effectively improve the user's satisfaction with the recommended item from the platform.

The vast majority of recommendation algorithms are based on two dimensions: user and item, which can be achieved through the explicit user feedbacks on the items, such as user ratings. Classical algorithms include content-based recommendation algorithms and collaborative filtering-based recommendation algorithms. The former recognizes the common characteristics of the items that have gotten the user's praise, then recommends other items to the user that have these common characteristics [2]. The latter is mainly based on the basic cognitive assumption that a user's evaluation of an item may be similar to that of other users on this item. So the target user's rating of this item is estimated by other users' ratings. This kind of algorithm overcomes the disadvantages of the content-based recommendation algorithm, and can recommend some items with different characteristics for users [3-4].

However, in many actual scenarios, users rarely express their explicit behavior, instead of implicit behavior, such as click and browse. At the same time, the recommendation algorithms based on explicit feedback mostly ignore the effect of the recommended context. Therefore, the research on recommendation algorithms based on implicit feedback has become a new hot topic in recent years. In [5], the authors mainly focus on the effect of context information and propose a context-dependent pre-filtering technique based on implicit user feedback. This technique associates user feedback with specific environment variables, cutting the recommendation system according to each specific context environment. In [6], a multi-layer context graph from implicit feedback data is abstracted and a context-dependent sorting algorithm is designed. In [7], the depth features of item contents are firstly extracted from the collected implicit feedback data, then the depth features are introduced into the Bayesian framework of pairwise queuing model, finally a deep collaborative filtering based on depth ordering technology is proposed.

Through the analysis of the above literatures, we can see that the main factors that affect the effectiveness of the recommendation system are not only the user ratings of the item, but also the users' concern about the item and contextual environment information (such as time and place). For the software sharing platform, the standard of what software should be recommended is the users' interest and recognition of the software, that is, how many users have accepted the software, which can help us to know how the users views the software. According to the relevant theories of social psychology research, a user's interest is easily influenced by the external environment and group behavior. The herd mentality is one kind of widespread psychology phenomena in society. Therefore, another area related to this study is the analysis and application of item popularity.

In the recommendation system, the popularity of an item refers to the amount of feedback that the item has received. The bigger the amount of feedback is, the higher popularity of item is [9]. Many popularity-based recommendation algorithms are very simple, such as major news and Weibo ranking list. These algorithms recommend some items to users only based on the page views (PV), the unique visitors (UV), average daily PV or share rate, etc. For example, in [10], the authors measure item popularity according to user feedbacks, and propose a simple algorithm for selecting truly popular items for users. Good performance is the biggest advantage of this algorithm. The investigation and study in [11] show that item popularity contains the rule of long-tailed distribution, so that the unpopular item recommendation from the long tail effect of item popularity distribution is very practical. In [12], the authors argue that a good recommender system should take the advantage of the Internet to recommend to users something that may be of interest but not easily discoverable, and to increase recommended coverage and novelty, not just to improve accuracy. In order to improve the ability of recommendation system to mine unpopular projects that user may interest, the authors introduce item popularity into the traditional collaborative filtering algorithm and improve the influence of unpopular items in the process of recommendation. The paper [13] also focuses on how to improve the recommendation of items with low popularity in sparse data environments and proposes an item-based collaborative filtering algorithm that considers the user activity and item popularity. When measuring the relativity of two items, the activity of the rated users and the popularity of the evaluated items are used to do a relevant punishment so as to improve the recommend probability of the items with low popularity.

In fact, the accuracy and the novelty are two conflicting goals in the recommendation algorithms. To solve this problem, in [14-15], the authors respectively balance the conflicting target through popularity prediction and normalized popularization to improve the accuracy and diversity of their algorithms. In addition, in [16-17], the authors considers the influence of time factor on the evaluation of item popularity, extracts the popularity feature and time continuity feature of the data to be evaluated, and

then proposes a temporal matching method to evaluate the popularity of interest points. Further, the studied that how to predict item popularity without user feedback or less user feedback in [18]. The authors use a resampling strategy to bias users towards these rare cases of highly popular items. The results of the above literatures show that the key to the research on item popularity is to analyze the data that may be relevant to the popularity in the item platform.

The main difference between the issues studied in this paper and the above literatures is that we are not recommending individual users with specific preferences, but rather recommending software for the general public. Unlike general products or services, software recommendation is a very special task as people use software for different reasons [19]. Therefore, the classical recommendation algorithm is not suitable for software recommendation on the Internet platform. On the one hand, due to the particularity of the software itself, it is very difficult to analyze the characteristics of the software and the content that can be analyzed is also very limited. On the other hand, a user's needs for software are often varied, so classic recommendation methods often fail to recommend items of different characteristics to the user. At present, there are few researches on software recommendation system, which is also the reason of this work. We will analyze the data in software-sharing Internet platform that affecting the software popularity, and then establish a recommendation method based on software evaluation model.

III. EVALUATION MODEL BASED ON DEGREE OF ATTENTION AND DEGREE OF SATISFACTION

A. Related concepts in the model

In order to facilitate the description of the problem, the relevant concepts involved in the evaluation model are given below.

Def.1 Software Views L The number of times a software have been viewed by users in the sharing platform since the software has been released.

Def.2 Software Downloads D The number of times a software have been downloaded by users in the sharing platform since the software has been released.

Def.3 Software Release Days T The number of days the software has been released.

Def.4 Software Evaluations P The number of times a software have been evaluated by users since the software has been released.

Def.5 High Ratings H The number of high ratings in P.

Def.6 Low Ratings C The number of low ratings in P.

The relationship between P, H and C satisfied:

$$P=H+C.$$

Def.7 High Rating Rate HP The ratio of favorable comments H to software evaluations P, namely H/P.

B. Influencing factors in the model

1) The degree of attention

In the Internet software sharing platform, the two key indicators that affect the popularity of software are L and D, which represents the user's attention to the software. The higher L and D are, the more popular the software is. However, software update is fast. If only L and D are recommended standards, some commonly used software will always be recommended. Therefore, the date of software release is also an important index to consider software recommendation. If user ratings are used as a recommended indicator, the newly released software should get a better rating. Therefore, L and D is proportional to the degree of attention, and inversely proportional to T. Based on the above analysis, we propose a formula for the degree of attention denoted by S_A, is as follows:

$$S_A = \frac{L+\beta D}{(T+1)^K}\ , \qquad (1)$$

where β is for the equivalent factor (β>1), K is for the control parameter. For users, the download behavior means greater interest in the software than the browsing behavior. As a result, the downloading behavior has more influence on the degree of attention than the browsing behavior.

2) The degree of satisfaction

L and D just represents users' interest in the software, and cannot represent the quality of software. Therefore, the recommendation method also need to consider the user feedback after using the software. However, the software as a special item, rating-based user feedback lacks a uniform standard, so rating is often too subjective. In the actual software sharing platform, only two evaluation criteria are generally set, such as " upvote" and "downvote", "like" or "dislike", and so on. Assuming a satisfactory feedback score is 1 and an unsatisfied feedback score is -1, the degree of satisfaction can be expressed as the difference between H and C. But for a software with H=1 and C=0 and another software with H=100 and C=1, this simple calculation would make the former more satisfied than the latter, which is obviously unfair. Therefore, it is very necessary to add P to the degree of satisfaction. The use of logarithm to calculate the degree of satisfaction can ensure that the software can also get a more fair evaluation in the early release. Based on the above considerations, we propose a formula for the degree of satisfaction, denoted by S_M, is as follows:

$$S_M = \log_n(H-C+P/10)\ , \qquad (2)$$

where n is for the control parameter. The smaller n is, the greater the influence of H on S_M is.

3) Wilson interval correction on HP

In most cases, P is not big enough, which leads to the evaluation of the software is not objective enough. Therefore, an algorithm is needed to correct the accuracy problem caused by the sufficient number of samples. Because if P is large, HP will be extremely objective for software. The problem is that it will go wrong P is very small. It is assumed that the software named as A has 3 high ratings and 0 low rating, while the software named as B has 100 high ratings and 2 low ratings. In this case, HP of A is higher than that of B. Obviously, the evaluation method is not reasonable and accurate. In the scenario studied in this paper, user feedback on software has only two kinds of values, so user feedback is statistically obeying the binomial distribution. The higher HP of the software means the more popularity it is. However, the credibility of HP depends on the number of users who have given high rating to the software. If the number of users is too small, the credibility HP is lower. We will calculate the HP's confidence interval to correct the impact of the small sample size on the evaluation on software popularity. Based on Wilson interval, we propose a new formula of HP is given [21]:

$$HP' = \frac{HP + \frac{1}{2P} z^2_{1-\frac{\alpha}{2}} - z_{1-\frac{\alpha}{2}} \sqrt{\frac{HP(1-HP)}{P} + \frac{z^2_{1-\frac{\alpha}{2}}}{4n^2}}}{1 + \frac{1}{P} z^2_{1-\frac{\alpha}{2}}} \quad (3)$$

where HP' is for the revised HP, $z^2_{1-\frac{\alpha}{2}}$ is for the z-statistic for the corresponding 95% confidence level.

4) Newton cooling correction on S_M

In the software sharing platform, new software always emerges. However, the degree of satisfaction of existing software does not automatically decrease over time, which makes the newly released software difficult to be effectively recommended. But in fact, a new software may have better performance, more worthy of being recommended. In this paper, we construct a decay function to simulate the natural cooling process of software popularity. That is, with the increase of T, in P and other parameters unchanged circumstances, S_M of the software will gradually declined. In this paper, we use Newton's law of cooling to establish the relationship between S_M and T. Newton's law of cooling states the law that is followed by an object that is hotter than the surrounding environment to transmit heat to the surrounding medium. The law notes that the speed at which the object cools is directly proportional to the difference in the temperatures between itself and its surroundings [20]. Assuming that the final S_M of all software approaches 0, the decline rate of S_M is directly proportional to the current value of S_M. We propose a specific cooling formula is as follows:

$$S_M^T = S_M^{T_0} e^{-\alpha \Delta T} \quad (4)$$

where $S_M^{T_0}$ is for S_M on day T_0, S_M^T is for S_M on day T, $\Delta T = T - T_0$, α is the cooling coefficient.

C. Overall evaluation model

The technical framework of the overall evaluation model based on software popularity is shown in Figure 1. The model contains two evaluation indicators and two correction formulas. First, we modify HP through the Wilson interval to make up for the impact of HP on the evaluation of software popularity due to the less P. Then, S_M is revised by Newton's cooling law, which helps us to overcome the problem that a new software which cannot be unfairly recommended. Finally, we calculate S_A and S_M, respectively, to generate software popularity that can be regarded as a basis for the recommended software.

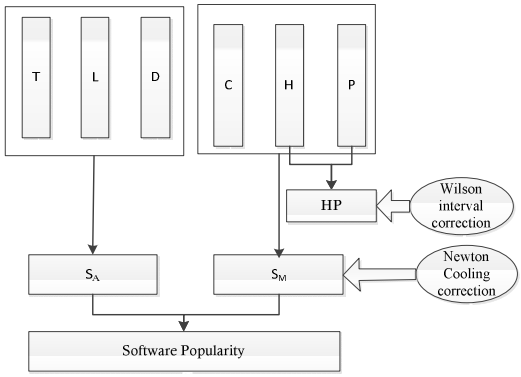


Fig.1 The technical framework of overall evaluation model

D. Software popularity recommendation algorithm

Algorithm 1 Software popularity algorithm based on evaluation model(SPAEM)

Input: L , D, T, P, H, C;
Output: Software popularity;
Step1. Calculate S_A according to formula (1) ;
Step2. Calculate S_M according to formula (2) ;
Step3. Wilson interval correction on HP according to formula (3);
Step4. Newton cooling correction on S_M according to formula (4);
Step5. Calculate software popularity, the formula is as follows:

$$S = \varepsilon_1 S_A + \varepsilon_2 \frac{HP'}{HP} S_M^{T_0} e^{-\alpha \Delta T} \quad (5)$$

where ε_1 and ε_2 is the weight coefficient.

IV. EXPERIMENTS AND ANALYSIS

In order to illustrate the implementation process of the software popularity recommendation method based on evaluation model, analyze and verify it, we set up the

simulation data to verify the inherent relationship among various factors in the proposed model, including:

(1) The relationship between the degree of attention and its influential factors;

(2) The relationship between the degree of satisfaction and its influential factors;

(3) The impact of correction formulas on the evaluation of software popularity.

In addition, we collect the actual data from the real platform to compare the proposed method with the recommendation results from other list of software.

The value range of the simulation data are shown in Table I. In addition, we also grabbed 9 days of data from the pea pod platform (https:// www.wandoujia.com/) as the basis of comparative experiments.

TABLE I. EXPERIMENTAL PARAMETERS SET

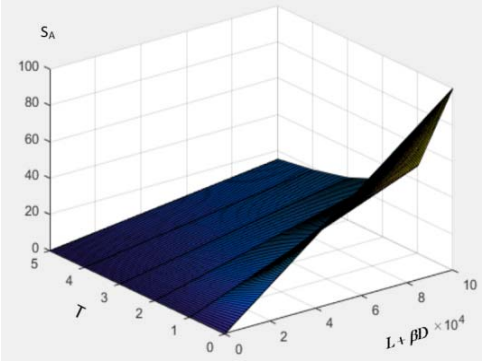
name	Ranges
L	1000～10000
D	L*0.1～L*0.2
T	1～100
P	D*0.1～D*0.3
H	0～P
C	P-H
β	10
ε_1	1
ε_2	1

A. *The relationship between the degree of attention and its influential factors*

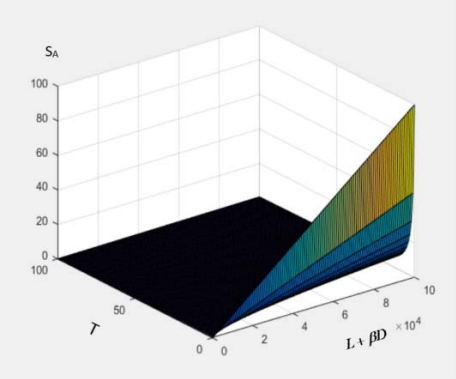
We simulate the process that users browse or download the software after it's released. The experimental results through the evaluation method based on the degree of attention are shown in Figure 2.

As can be seen from Figure 2, at the beginning of the software release, with growth of L and D, S_A shows a trend of slowly increasing and then rapidly increasing. That is, if L and D achieve fast growth in a short time, S_A will be significantly improved. However, as T increases, the inflection point of S_A increases gradually. As the number of downloads and views of the software does not increase as the release time increases, the attention of the software will inevitably decrease.

In the following experiments, we analyze the control parameter K's impact on S_A by adjusting the value of K if D and L don't change. Set K to 1.2, 1.5 and 2.5, respectively. The experimental results are shown in Figure 3. As can be seen from figure 3, the larger the value of K, the faster the degree of attention declines, that is, the value of K determines the update speed of recommendation software. If you hope that a software whose D and L have just a smaller changes in a certain period of time is quickly eliminated, you should set a larger value for K, and vice versa.



(a) The Changes of S_A (T = 5)



(b) The Changes of S_A (T = 100)

Figure 2 The relationship between L,T,D and S_A

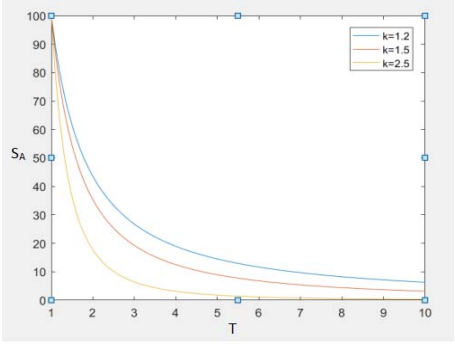


Figure 3 The influence of K on S_A

B. *The Relationship between the degree of satisfaction and its influential factors*

In this experiment, number the software's IDs from 1 to 10, and then random generate respective H and C. The experimental results from the evaluation method of S_M are shown in Table Ⅱ. As can be seen from Table Ⅱ, when C is obviously greater than H, S_M is negative; when C is approximately equal to H, the bigger the value of P, the greater the degree of software satisfaction, such as software 1 and software 9; when the values of H of two

software are approximately equal, their difference of C has less influence on S_M than the difference itself, such as software 2 and software 10. It means that as long as a software has enough high ratings, in theory, the software is worth recommending, but if a software's low ratings are far more than its high rating, the software would not be recommended.

TABLE II. THE RELATIONSHIP BETWEEN H, C, P AND S_M

ID	H	C	P	S_M
1	165	129	294	6.03
2	563	200	763	8.78
3	779	198	977	9.41
4	472	259	731	8.16
5	349	169	518	7.86
6	605	127	732	9.11
7	658	329	987	8.74
8	6	31	37	-2.76
9	45	35	80	4.17
10	560	100	660	9.04

C. *The Impact of Correction Formulas on Software Popularity Evaluation*

In this experiment, we use formulas (3) and (4) respectively to correct HP and S_M for software in Table Ⅱ. The results are shown in Table Ⅲ.

TABLE III. THE IMPACT OF CORRECTION FORMULAS ON SOFTWARE POPULARITY EVALUATION

ID	T	Amended HP	S_M	Amended S_M
1	12	0.50	6.03	3.31
2	24	0.71	8.78	2.64
3	9	0.77	9.41	6.00
4	10	0.61	8.16	4.95
5	21	0.63	7.86	2.75
6	3	0.80	9.11	7.84
7	17	0.63	8.74	3.74
8	21	0.08	-2.76	-0.96
9	1	0.45	4.17	3.97
10	22	0.82	9.04	3.01

Compare to Table Ⅱ, Wilson interval correction generally leads to a decrease of the original value of H. However, for the different P, the rate of decline is different. The less the P, the more obvious the decline. For example, after correction, the HP of software 1 dropped from 0.56 to 0.50, while the HP of software 9 dropped from 0.56 to 0.45. Obviously, the HP of software 1 is more convincing than that of software 9. For Newton cooling correction, we can see through comparing the results of Table Ⅱ and Table Ⅲ that the software with short release time are less correct than the software with long release time, especially for the latter without obvious change of T and L, the degree of satisfaction will be significantly reduced, such as software 2 and software 10.

D. *Comparison of different methods of software popularity evaluation*

Because most of the platform has no public statistics for the evaluation indexes proposed in this paper, such as L. In this experiment, we assume that D is in proportion to L. In this case, the impact of D on software popularity is equal to L+βD. We compare a public ranking list of the software with the method proposed in this paper to analyze the reasonable about the final recommendation ranking. The actual data of the evaluated software platform in the experiment is shown in Table Ⅳ, and the experimental result is shown in Figure 4.

TABLE IV. RELATED DATA ON EVALUATED SOFTWARE PLATFORM

ID	L&D	H/P	P
1	27930000	52%	17108
2	11760000	77%	15597
3	5750000	64%	6831
4	4870000	91%	5167
5	4080000	71%	6608

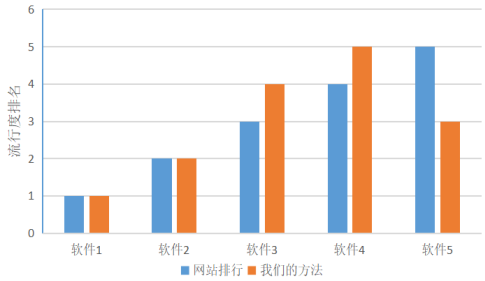


Figure 4 Comparison of the software popularity evaluation methods

From the ranking results in Figure 4, we can see that our result is different from that of the popularity evaluation method in the website. The main difference is the ranking of software 3, software 4, and software 5. It can be seen that the original ranking in the website mainly depends on the value of D. And for our method, software 1 and software 2 gained high ratings due to their huge downloads. Although the D&L of software 4 is smaller than that of the software 3, the H of software 4 is far greater than that of the software 3. In this case, the impact of user experience on software

popularity is obviously greater than simple download behavior. Similarly, although the D&L is greater than that software 3, after comprehensively considering S_A and S_M, software 5 gains higher order than software 3 due to its more high ratings.

In summary, the software popularity recommendation method mentioned in this paper is more fully considered the factors that affect the recommendation effect, which can get more reasonable recommendation results.

V. CONCLUSION

Aiming at the problem of software recommendation in the Internet platform, this paper proposes a software popularity recommendation method based on evaluation model. The main contributions include three aspects: 1) The various factors which can affect the recommendation effect of software sharing platform are analyze. And then the internal relations between these factors are established. 2) The problem of the impact of the small sample data on the accuracy of recommendation, to a certain extent, is solved by setting the confidence interval; 3) The problem of cold start of newly released software, to a certain extent, has been solved by means of automatic decay of popularity. Finally, the validity of the software popularity recommendation method is verified by using the example analysis method. In the future work, we will further analyze the emotional factors in the user evaluations to construct the user preference label, and then propose a personalized software recommendation model based on the user preference label.

ACKNOWLEDGMENT

The authors wish to thank Natural Science Foundation of China under Grant No.61662054, 61622082 and 61462066, Natural Science Foundation of Inner Mongolia under Grand No. 2015MS0608, Inner Mongolia Science and Technology Innovation Team of Cloud Computing and Software Engineering, Inner Mongolia Application Technology Research and Development Funding Project “Mutual Creation Service Platform Research and Development Based on Service Optimizing and Operation Integrating” and CERNET Network Next Generation Internet Technology Innovation Project under Grand No. NGII20160511.

REFERENCES

[1] Yan Z, Zhang P, Deng R H. TruBeRepec: a trust-behavior-based reputation and recommender system for mobile applications[J]. Personal and Ubiquitous Computing, 2012, 16(5):485-506.

[2] Pessemier T D, Vanhecke K, Dooms S, et al. Content-based Recommendation Algorithms on the Hadoop MapReduce Framework[C]// Webist 2011, Proceedings of the, International Conference on Web Information Systems and Technologies, Noordwijkerhout, the Netherlands, 6-9 May. DBLP, 2011:237-240.

[3] Desrosiers C, Karypis G. A Comprehensive Survey of Neighborhood-based Recommendation Methods[M]// Recommender Systems Handbook. Springer US, 2011:107-144.

[4] Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Transaction on Information Systems22(1), 143–177 (2004)

[5] Baltrunas L, Amatriain X. Towards time-dependant recommendation based on implicit feedback[J]. In Workshop on context-aware recommender systems CARSâ Ă´ Z09, 2009.

[6] Yao W, He J, Huang G, et al. A Graph-based model for context-aware recommendation using implicit feedback data[J]. World Wide Web, 2015, 18(5):1351-1371.

[7] Ying H, Chen L, Xiong Y, et al. Collaborative Deep Ranking: A Hybrid Pair-Wise Recommendation Algorithm with Implicit Feedback[C]// Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer International Publishing, 2016:555-567.

[8] Luo Z.M., Consumer psychology [M]. Tsinghua University Press Co., Ltd., 2002.

[9] Steck H. Item popularity and recommendation accuracy[C]// ACM Conference on Recommender Systems, Recsys 2011, Chicago, Il, Usa, October. DBLP, 2011:125-132.

[10] Vojnović M, Cruise J, Gunawardena D, et al. Ranking and Suggesting Popular Items[J]. IEEE Transactions on Knowledge & Data Engineering, 2009, 21(8):1133-1146.

[11] Yin G.S., Zhang Y.N., Dong H.B., et al.A recommendation method constrained by long tail distribution [J] .Journal of Computer Research and Development, 2013, 50 (9): 1814-1824.

[12] Hao L.Y., Wang J..Cooperative TopN Recommendation Algorithm Based on Item Popularity [J]. Computer Engineering and Design, 2013,34 (10): 3497-3501.

[13] Wang J.K., Jiang Y.C., Sun Q.S., et al.Research on collaborative filtering based on nearest neighbor of project considering user activity and project popularity [J] .Computer Science, 2016, 43 (12): 158-162.

[14] Javari A, Jalili M. Accurate and Novel Recommendations: An Algorithm Based on Popularity Forecasting [M]. ACM, 2015. ACM , 2015 , 5 (4) :1-20

[15] Zhao X, Chen W, Yang F, et al. Improving Diversity of User-Based Two-Step Recommendation Algorithm with Popularity Normalization[M]// Database Systems for Advanced Applications. Springer International Publishing, 2016.

[16] SI Y.L., L. F., Song Y.W..Recent interest point recommendation algorithm based on epidemic features and kernel density estimation [J] .Microcontroller Systems, 2016,37 (11): 2416-2420..

[17] Yao Z, Fu Y, Liu B, et al. POI Recommendation: A Temporal Matching between POI Popularity and User Regularity[C]// IEEE, International Conference on Data Mining. IEEE, 2017.

[18] Moniz N, Torgo L, Eirinaki M, et al. A Framework for Recommendation of Highly Popular News Lacking Social Feedback[J]. New Generation Computing, 2017:1-34.

[19] Liu Y, Du F, Jiang Y, et al. A Novel APPs Recommendation Algorithm Based on APPs Popularity and User Behaviors[C]// IEEE First International Conference on Data Science in Cyberspace. IEEE Computer Society, 2016:584-589.

[20] Evan M. Rank H.. With Newton's Law of Cooling [online], http://www.evanmiller. org/ rank-hotness-with-newtons-law-of-cooling.html, 2009.

[21] Evan M.. How Not To Sort By Average Rating [online].