The Analysis and Design of the Job Recommendation Model Based on GBRT and Time Factors

Pengyang Wang
Institute of Information Security, School of Computer Science
Beijing University of Posts and Telecommunication
Beijing, China
e-mail: merlinwang555@gmail.com

Yingtong Dou
International School
Beijing University of Posts and Telecommunication
Beijing, China
e-mail: douyingtong@bupt.edu.cn

Yang Xin
Institute of Information Security, School of Computer Science
Beijing University of Posts and Telecommunication
Beijing, China
e-mail: yangxin@bupt.edu.cn

***Abstract*—In this paper, we first give a comprehensive summary of models and algorithms applied in three online job recommender systems and point out the advantages and disadvantages of these models. Then we introduce a job recommendation model based on Gradient Boosting Regression Tree and time factors (T-GBRT). The T-GBRT model aggregates the time factors into the GBRT to predict personal preferences and adds time factor weight to topK rankings, with a neighbor based filtering trick in reducing the amount of calculation. At the end of the paper, the model performs the best in the experiment with four criterions, comparing to other three models, which proves the efficiency of the new model.**

Keywords-jobs recommendation, time factors, T-GBRT

I. INTRODUCTION

As an efficient way to deal with the information overloading problem [1], the recommender systems have been widely applied into many web applications. The application of recommender systems varies in different scenarios and jobs [2]. Taking the online shopping website as an example, the website need to give personalized recommendation to the customers considering the trend of the market as well as the customers' own tastes. The online recruitment is another scenario different with e-commerce. We need to consider whether the ability of job hunters is compatible with the corresponding job at first [3]. After that we focus on the job hunters' preference. The preference maybe the personalized requirement for working places and environment. So we need to consider both the users' preference and their ability when designing the recommender systems for online recruitment.

For the designing of recommender systems for online recruitment, there are two main models having been proposed.

Collaborative Filtering (CF) [4]. There are user based CF and item based CF [5] and being applied in the scenario where the number of users is bigger or the number of items is bigger respectively. R. Rather, K. Bradley and B. Smyth [6] had proposed two approaches in CASPER system. One method is based on the user based CF; it recommends target users the jobs which their similar users like. The measurement of similarity is based on the ratings of users to jobs. For the similar users, they calculate the similarity of user ability and job requirement and rank the jobs according to it. CASPER gives recommendation according to the ranking. Calculating the similarity of users and jobs to evaluate the compatibility of users and jobs; it fulfills the compatibility but without considering the users' preference. W. Hong et al. clustered the users into three clusters according to their information and take three different recommendation policies [7]. One of the method is based on user based CF.

Content based recommendation [8]. Its main idea is to calculate the jobs which are similar to the jobs having been interacting with users according to the users' interacting record and then recommend them to the users. The measurement of its similarity is based on the content of the job itself which has many types i.e. the texts. CASPER [6], the iHR of W. Hong [7] et al., and the PROACTIVE of Lee [9]. Since it has considered the properties of the jobs and the interacting records of users, the content based recommendation has a better performance than CF.

II. PREPARATION WORK

In this section, we present the preliminaries and the problem formulation of this study. We introduce the building process of regression trees based on CART algorithm, the gradient boosting algorithm and the measurement of similarity.

978-1-5090-3471-0/16/$31.00 ©2016 IEEE

A. Regression Tree

Linear regression is a regression model based on global data. The model building process is very complex for some big data. Tree regression is a local model building method. It divides the data set into different domains according to the decision tree then builds the model on the sub-divided domain[10]. We introduce the model building process taking CART algorithm as example.

1) Tree construction

Calculating the best division feature

a) Iterating all the features.
b) For each feature, iterating all the eigenvalues of it and calculating the error of using this value to divide.
c) Taking the feature with the minimum error as its best division feature and going back.

Finding the best division features.

a) Store the node as leaf node then come back if the node could not be divided anymore.
b) Dividing the data set into left-right subtrees according to the best division feature.
c) Repeating the above steps for left and right subtrees respectively.

2) Regression tree based prediction

When the regression tree is constructed, the next step is to predict on the new data. Here, the new data x, defined as $x=(x_1,\ldots, x_n)$, x_i is the *i*th feature. The node is defined as the current node of the tree, the left subtree of the node is node.left and the right subtree of the node is node.right. The detail about prediction is shown as TABLE I.

TABLE I. REGRESSION TREE BASED PREDICTION

Algorithm I. Regression tree based prediction
1: Input: x, node
2: index = node, $x_{cur} = x_{index}$
3: if node is root
4: return node as prediction
5: else
6: if $x_{cur} > x_{node}$
7: node = node.left, goto line 2
8: else
9: node = node.right, goto line 2
10: end if
11: end if

B. Gradient Boosting

Gradient boosting algorithm[11] is a strong regression algorithm integrated with many other regressors. Its main idea is to build a new regression tree in the descending direction of the loss function gradient based on the results of the last regression tree. The loss function describes the difference between prediction value and real value. We take it as the standard to evaluate the precision of prediction. The key approach is to increase the approximate value of the residual in the tree algorithm with the aid of the value of loss function negative gradient, then fitting a regression tree. The algorithm is shown below.

The input data is $T=\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, $x_i \in X \subseteq R^n$, $y_i \in Y \subseteq R$;The loss function is$L(y, f(x))$;

Intialize

$$f_0(x) = argmin_c L(y_i, c) \tag{1}$$

For $m=1, 2 \ldots, M$

a) For $i=1, 2, \ldots, N$,

$$r_{mi} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)} \tag{2}$$

b) For r_{mi}, fitting a regression tree

c) For j=1, 2, …, J,

$$c_{mj} = argmin_c \sum_{x_i \in R_{mj}} L(y_i, f_0(x_i) + c) \tag{3}$$

a) Update

$$f_m(x) = f_{m-1}(x) + \sum_{j=1}^{j} c_{mj} I(x_i \in R_{mj}) \tag{4}$$

Get the regression tree

$$\hat{f}(x) = f(x) = \sum_{m=1}^{M} \sum_{j=1}^{J} c_{mj} I(x_i \in R_{mj}) \tag{5}$$

The first step is initialization, estimating the constant which makes the loss function reach its minimum value. It is a tree with only one root node. Eq. (2) calculates the value of the loss function negative gradient in the current model. It will be taken as the estimation of the residual. Eq. (3) estimates the node area of the regression tree to fit the estimated value of residual. Eq. (4) estimates the value of the leaf node area with linear searching. To minimize the value of loss function, Eq. (5) refreshes the regression tree. The third step produces the final model $\hat{f}(x)$.

C. The Measurement of Similarity

The compatibility of job hunters' abilities and background to the job requirement must be considered at first in the recruitment. In this scenario, the compatibility could be regarded as the similarity between job hunters' information and jobs' information. The compatibility increases with the increment of similarity. The information of job hunters and the job are both text information. In information retrieval, the similarity of two text can be obtained by calculating the similarity of text vector in vector space model[12]. We extract key words from the information of job hunters and jobs and transfer them into vectors. Estimating the compatibility between job hunters and jobs by calculating the similarity between vectors.

Let $User_i=(u_{1,i}, u_{2,i}, \ldots, u_{n,i})$ represents job hunter i and $Item_j=(i_{1,j}, i_{2,j}, \ldots, i_{n,j})$ represents job j. Taking cosine similarity to represent their similarity.

$$Similarity_{A,B} = \frac{User_i \cdot Item_j}{\left\|User_i\right\| \left\|Item_j\right\|} = \frac{\sum_{i=1}^{n} (u_{t,i} \cdot i_{t,j})}{\sqrt{\sum_{i=1}^{n} u_{t,j}^{\ 2}} \cdot \sqrt{\sum_{i=1}^{n} i_{t,j}^{\ 2}}} \quad (6)$$

The degree of the vector is *n*.

In a similar way, the similarity between job hunters and the similarity between jobs could also be calculated with above method.

III. JOB RECOMMENDATION MODEL BASED ON GBRT AND TIME FACTORS

Our model includes four main parts, the feature extraction, the chosen of alternative set, the training and prediction of GBRT (Gradient Boosted Regression Trees) model and the TopN recommendation generated by the weight of time factor. The model is shown in the Fig. 1.

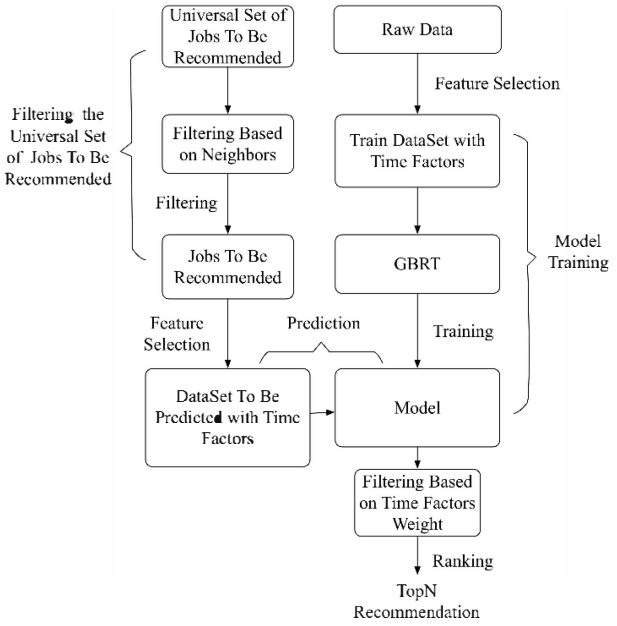

Figure 1. This figure introduces details the structure of the T-GBRT model.

We introduce the concept of time factor to the model. Time, which is an important context, has impressive influence on users' preference and has an effect on the recommendation result. The effects of time to users' preference are mainly in three types shown in below.

The preference varies with the time. As the age growing and having more experience, the user's preference also changes. In the job hunting scenario, the fresh graduates are interested in primary positions and the job hunters with more than ten years' working experience will show his/her interest in senior positions like managers. Besides, the job seeking experience also have a great effect on the job hunter's job preference and this effect is cumulative.

The life cycle of the job itself. A job has a life cycle in the recruitment and the system could only recommend corresponding jobs to the job hunters in a specified period. Though the job hunters have shown their job preferences explicitly or inexplicitly, it makes no sense recommending invalid jobs to job hunters

The seasonality of recruitment. The recruitments are usually classified to campus recruitment and social recruitment. Campus recruitment has seasonality. Taking IT companies as example, the autumn of every year is the campus recruitment season for fresh graduates and there will be a lot of new jobs. The seasonality is an important factor in job recommender systems design.

There have been some researches about the time factors in recommender system. Zhen et al. [13] add linear attenuation to the history rating data to show the trend of users' preference. Yucai Zhou et al. [14] propose a hybrid recommendation model based on time factor aiming to improve the accuracy of user similarity calculations.

This paper considers the effect of time factors with our GBRT model.

A. Feature Selection

We define the job recommender problem as a prediction problem. Giving different types of interaction corresponding marks according to users' interacting history, we predict the marks user might be given in the future to one specific job. In the Internet, different interacting types means the degree a user likes different items. The usual interacting types are clicking, deleting, favorite and submitting resume etc. We could rank each interacting type according to their representation users' preferences. Since P (Submitting) > P (Favorite) > P (Clicking) > P (Deleting), for giving marks to each behavior, R (Submitting) > R (Favorite) > R (Clicking) > R (Deleting). Submitting resume, favorite and clicking is the positive interaction and deleting is the negative interaction. P represents the degree of preference and R represents rating.

Since there are three dimensions (job hunter, job and time) in job recommender system, we classify the feature into user feature, job feature and time feature in our model.

1) Job Feature

Different jobs have different click rate; the rate reflects the popularity of the job. There are positive and negative interaction and they are calculated by the times of positive or negative interaction and the number of users on the platform respectively as the description of the feature of the job on this platform. They are:

The number of users having positive interaction with the job.

The number of users having negative interaction with the job.

The times of job be interacted positively.

The times of job be interacted negatively.

In addition, the attribute of the job itself can be generated by clustering.

2) Job Hunter Featrue

The feature of job hunter reflects his/her custom and preference. We define two job hunter features which are self attribute and personal preference. Personal preference is

divided into general preference and special preference which is related to the job.

a) Personal preference feature

General preference feature. General preference feature describes the preference degree of users to the stuffs not related to the job or platform. It is used to describe the users' preference to the online job hunting. They are extracted from the user's behavior log. They are:

- The user's online time so far
- The number of jobs with positive interaction to user so far.
- The number of jobs with negative interaction to user so far.

Special preference feature. Special preference feature describes the preference a user to specific job. It is expressed by the interaction history of the user to a job.

b) The group feature

The group feature is represented by the user's class. If there is no class, the class is produced by clustering.

3) The time feature

Our model predicts the future interaction rating based on the history data, thus we add the time feature into our model. The granularity of time could be days, weeks, months or years and it varies according to the specific data.

4) Time factor based feature

The future interaction will be influenced by the past behavior online. Taking online shopping as example, if the user bought item A and liked A, the user will show his/her preference to B when the user buys item B which is similar to A after a month. It could be seemed as the influence from A to B. We need to consider the attenuation effect and aggregation effect when we describe the feature.

The attenuation effect means that the interaction which is longer to now has less influence on the present. If the influence is represented by function F, F is smaller while the time duration is larger.

The aggregation effect means the jobs having been interacted with the user all have influence on the further interaction. We need to accumulate the influence of all the interacted jobs when we quantify the aggregation effect.

The time factor based feature is defined as

$$feature_{time\_factor}(u,j)=\sum_{\Delta t}F(\Delta t)(\sum_{i}G(w_{i,j},r_{u,j})) \tag{7}$$

u means the specific user, i means the jobs having interacted with user u. $w_{i,j}$means the similarity between job i and job j, $r_{u,i}$ means the rating of user u to job i. Δt means the time duration from the time when user u has interaction with job i to now. $F(\Delta t)$ represents the time attenuation effect and it decreases as the Δt increases. $G(w_{i,j}, r_{u,i})$ represents the influence from once interaction between user u and job i to user u and job i without consideration of time factor.

The user u has no direct relation to job j but they are both related to the job i. The relation between user u and job i is the rating $r_{u,I}$ and $w_{i,j}$ connects job j with job i. The job i is the bridge between user u and job j. Their relationship can be shown as Fig. 2.

For (7), a simple definition is

$$feature_{time\_factor}(u,j)=\sum_{\Delta t}(\frac{1}{\Delta t+1})(\sum_{i}w_{i,j}\times r_{u,j}) \tag{8}$$

Here, $F(\Delta t)=\frac{1}{\Delta t+1}$, $G(w_{i,j},r_{u,i})=w_{i,j}\times r_{u,I}$, which fulfill the requirement of (7).

5) The similarity between job hunters and jobs

The description of job hunters and jobs are very complex. We could mine the information from them by formatting them into text vectors. We calculate the similarity between job hunters and jobs by calculating the vector similarity. The similarity could be described as the compatibility between jobs and hunters. The details are shown in equation (6).

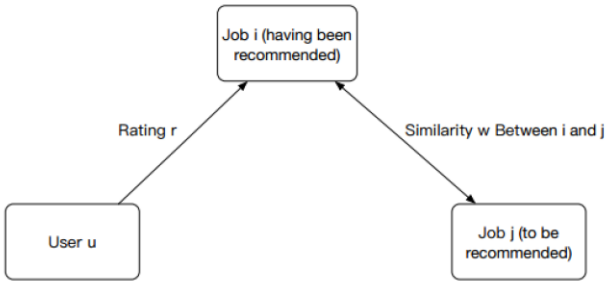

Figure 2. This figure describe the relationship between user u, job i (having been recommended) and job j(to be commended).

B. Filtering the Set of Awaiting Recommended Jobs

In online job hunting website, there are millions of jobs to choose and they have been updating every day. If we predict the preference of each user to every jobs directly, the amount of calculation will very large. To reduce the amount of calculation, we propose a filtering method which based on neighbor domain to reduce the set of awaiting recommended jobs to a reasonable scale. We collect the job A which has been interacted with user from a user's behavior log and represent it as a vector A=(a_1, a_2, …, a_n). Then we choose the awaiting recommended jobs from the awaiting set and represent them as vector B=(b_1, b_2, …, b_n). The similarity of AB is

$$Similarity_{A,B}=\frac{A\cdot B}{\|A\|\ \|B\|}=\frac{\sum_{i=1}^{n}(a_i\times b_i)}{\sqrt{\sum_{i=1}^{n}a^2}\times\sqrt{\sum_{i=1}^{n}b^2}} \tag{9}$$

Ranking the positions in the descending sequence, we take the top K jobs as the new awaiting set. This method will be validated in the experiment part.

C. Other Parts of the Model

The traning and Prediction of GBRT model

We preprocess the data as the methods above and input them as training data to the GBRT. The model after training will be used to predict marks.

Time factor based filtering

We multiply the marks and time factor as the final result. Ranking the final results in descending order, we recommend the top K jobs to the job hunters.

IV. EXPERIMENT

A. Description of Data and Tools

The data in the experiment are from the 2016 ACM Conference on Recommender Systems challenge. The data are provided from the sponsor www.xing.com which is a job hunting website. The data has four main parts, the recommendation generated by the website users themselves (impression.csv), the user information (users.csv), the job information (items.csv) and the user behavior log (interactions.csv). The time span is from the 34th week to the 46th week in 2015. We choose the data from the 34th to the 45th week. The training set is the data from the 34th to the 44th week. The data of the 45th week is the offline testing data. Here, in the training process, the number of total job hunters is 741650 and the number of jobs is 1500000; in the test process, the number of target job hunters is 133796 and the scale of available awaiting job lists is 370925.

The tools used in the experiment is provided by Graphlab, which is a parallel scientific computation tool. Here, we use the official implementation of the GBRT algorithm by Graphlab; IBCF, UBCF and CBR is implemented by the Graphlab package.

B. Evaluation Method

The testing method refers to the testing method of 2016 ACM Conference on Recommender Systems challenge. We calculate the marks based on the scoring function score(S, T). Here, S is the solution which is generated by the T-GBRT and T is the test set that describes a list of jobs having interaction with the user in the next week. S and T are all composed by the 2-tuple set with user and his/her corresponding job list. The function score(S, T) refers to 3 other supplement functions, respectively precisionAtK, recall and userSuccess. The precisionAtK describes the precision within the first topk items, the recall function describes the recall of the model and the user success illustrates the level of recommendation success. The details about functions show as TABLE II~V.

TABLE II. FUNCTION SCORE

Function I. score(S, T)
1: score = 0.0 2: foreach(u, t) in T: 3: r = S(u) 4: score += 20*(precisionAtK(r. T, 2) + precisionAtK(r. T, 4) + precisionAtK(r. T, 6) + precisionAtK(r. T, 20)) 5: return score

TABLE III. FUNCTION PRECISIONATK

Function II. precisionAtK(r, T, k)
1: topK ← get the first K of the recommendation list 2: the length of intersection of topK and relevantItems / k

TABLE IV. FUNCTION RECALL

Function III. recall(recommendedItems, relevantItems)
1: if (recommendedItems, relevantItems) 2: return intersect(recommendedItems.take(30), relevantItems.size) / relevantItems.size 3: else 4: return 0.0

TABLE V. FUNCTION USER SUCCESS

Function IV. userSuccess(recommendedItems, relevantItems)
1: if (intersect(recommendedItems.take(30), relevantItems) > 0) 2: return 1.0 3: else: 4: return 0.0

TABLE VI. THE DEFINITION OF USER RATINGS

Interaction Type	Score
Click	1
Collect	2
Delivery Resume	3
Delete	-4

TABLE VII. DESCRIPTION OF SELECTED FEATURES

Feature Name	Description	Remarks
user_num_weeks	User's the number of weeks on the platform	User feature(common feature)
user_num_act	User's The number of active interaction	User feature(common feature)
user_num_neg	The number of negative interaction	User feature(common feature)
user_num_last_rec	The times of prediction on the job last week	User feature(special feature)
user_num_past_rec	The times of prediction on the job last week in the past	User feature(special feature)
user_is_neg	Whether negative interaction occurred between the job and the user	User feature(special feature)
user_num_item_last	The times of interaction between the user and the job	User feature(special feature)

user_category	The category of the user	User feature
item_num_act	The number of active interaction of the job	Job feature
item_num_neg	The number of negative interaction of the job	Job feature
item_num_user_act	The number of users who have interacted actively with the job	Job feature
item_num_user_neg	The number of users who have interacted negatively with the job	Job feature
item_category	The category of the item	Job feature
past_time_weight	The past effect of interaction on the job	Time factor based feature
last_time_weight	The last-week effect of interaction on the job	Time factor based feature
week	time	Time feature
user_item_similarity	The similarity between the user and the item	

TABLE VIII. PARAMETER OF GBRT MODEL

The Number of Trees	The Max Depth	RMSE
8	5	1.6915

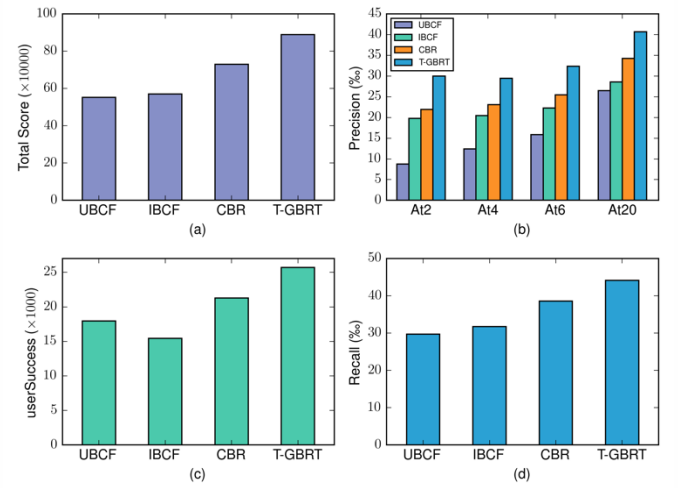

Figure 3. (a) illustrates the total score of the four model respectively, according to the evaluation method above. (b) gives the precision of four models at the first 2, the first 4, the first 6 and the first 20, respectively. (c) gives the number of success recommendation of four models. (d) gives the recall of four models. (a)~(d) correspond to the numerical values in the TABLE IX, respectively.

C. *The Definition of User Ratings*

The definition of user rating is shown in the TABLE VI.

D. *The Set of Features*

The features selected in our experiment and their notation is shown in the TABLE VII.

Since there is no class of users and items provided by the dataset, we clustered 20 classes from users and items respectively, according to their keywords.

E. *The Parameter of GBRT Model*

The parameters of GBRT model generated by training is shown in the TABLE VIII.

F. *The Comparison of Results*

To show the validation of our model, we compare T-GBRT with user based collaborative filtering (UBCF), item based collaborative filtering (IBCF) and recommendation based on content with the same data set mentioned above. The result of the numerical value is shown in TABLE IX. in detail.

1) The final result comparison

According to the Fig. 3(a), the total mark of each model is GBRT based on time factor (T-GBRT)(888730) > Content based recommendation (CBR) (728880) > IBCF (569480) > UBCF (551550). The IBCF performs better than UBCF. The reason is that he the number of items is approximately four times of the number of users in the dataset. So the item based collaborative filtering has more advantages. T-GBRT ranks at top 1 and CBR follows. The reason is that the model we propose in the paper learns the user's latent preference with machine learning method and combines the time factor into the model. It is better than the CBR which is memory based.

2) The precision comparison

From Fig. 3(b), T-GBRT performs better than other models in the top 2, top 4, top 6 and top 20 precision especially the first two precisions. It shows that the precision of the model we proposed is better at the first half part.

3) The recall comparison

From Fig. 3(c), T-GBRT performs best and then CBR, IBCF and UBCF. It means that the model we proposed in the paper has better coverage in the target jobs and is easier to discover the user's preferences.

TABLE IX. PARAMETER OF GBRT MODEL

	UCBF	ICBF	CBR	T-GBRT
Total Score	551550	569480	728880	888730
Precision_At2	8.77‰	19.79‰	21.96‰	29.99‰
Precision_At4	12.40‰	20.42‰	23.11‰	29.42‰
Precision_At6	15.84‰	22.25‰	25.43‰	32.33‰
Precision_At20	26.49‰	28.54‰	34.28‰	40.70‰
userSuccess	17941	15451	21257	25696
Recall	29.69‰	31.73‰	38.59‰	44.15‰

4) The number of success comparison

The Fig. 3(d) shows the number of the success in the recommendation. The result is T-GBRT (25969) > CBR (21250) > UBCF (17941) > IBCF (15451). Comparing with other criterion, T-GBRT performs not that better than others. It means that the T-GBRT performs better in one-user problem. Meanwhile, the IBCF performs better in other criterions, so the IBCF performs more precisely and has better coverage than UBCF in one-user problem.

5) The conclusion of experiment results

From the comparison above, we could get that the T-GBRT is better than other three models in total score, precision, recall rate and hit numbers. For the precision, the improvement of TopN at the first half part and last half part is quite a lot and it shows that our model could optimize the user experience.

UBCF and IBCF only consider the interaction between users and items but not the influence of content and time. CBR add the content but not the time factor. Our T-GBRT model integrate the interaction history, neighbor, time and performs better in each criterion.

V. CONCLUSION

The paper proposes the GBRT model based on time factor. We add time factor to features and take neighbor domain based filtering for job filtering to reduce the amount of calculation. We also rank the output of GBRT model combined with time factor to get the TopN recommendation list. Then we prove the validation of our model by experiment.

The most innovative issue in the model is the time-factors applied in the feature selection and the final ranking step. During the feature selection, the introduction of time factors enhance the capability of the model to look backward to the history of the user, accounting for the inexplicit preference of pastime. As for the ranking step, time-factor weight, integrating with the output of the gbrt model, performs better than that only the output of the gbrt model as the final result. All of these illustrate the validation of the introduction of the time factors.

ACKNOWLEDGEMENTS

This work was supported in part by the Institute of Information Security, with graduate grant from the Department of Information Security, School of Computer Science, Beijing University of Posts and Telecommunication, Beijing, China.

Thanks to the open data provided by ACM sponsor XING and the parallel machine learning platform GRAPHLAB, introduced by DATO which is a IT company focusing on specific computing tools. With the help of data and tools, we successfully prove our ideas.

REFERENCE

[1] M. J. Eppler and J. Mengis, "The Concept of Information Overload - A Review of Literature from Organization Science, Accounting, Marketing, MIS, and Related Disciplines.," *Inf. Soc.*, vol. 20, no. 5, pp. 325–344, 2004.

[2] D. H. Park, H. K. Kim, Il Young Choi, and J. K. Kim, "A literature review and classification of recommender systems research.," *Expert Syst. Appl.*, vol. 39, no. 11, pp. 10059–10072, 2012.

[3] A. Gupta and D. G. 0002, "Applying data mining techniques in job recommender system for considering candidate job preferences.," *ICACCI*, pp. 1458–1465, 2014.

[4] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "GroupLens - An Open Architecture for Collaborative Filtering of Netnews.," *CSCW*, pp. 175–186, 1994.

[5] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms.," *WWW*, pp. 285–295, 2001.

[6] R. Rafter, K. Bradley, and B. Smyth, "Personalised retrieval for online recruitment services," *In Proceedings of the 22nd Annual Colloquium on Information Retrieval*, 2000.

[7] W. Hong, S. Zheng, H. Wang, and J. Shi, "A Job Recommender System Based on User Clustering.," *JCP*, vol. 8, no. 8, 2013.

[8] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems.," *The Adaptive Web*, vol. 4321, no. 10, pp. 325–341, 2007.

[9] D. H. Lee and P. Brusilovsky, "Fighting Information Overflow with Personalized Comprehensive Information Access: A Proactive Job Recommender," presented at the Third International Conference on Autonomic and Autonomous Systems (ICAS'07), 2007, pp. 21–21.

[10] L. Olshen and C. J. Stone, "Classification and regression trees," *Wadsworth International Group*, 1984.

[11] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, 2001.

[12] D. L. Lee, H. Chuang, and K. Seamons, "Document ranking and the vector-space model," *IEEE software*, 1997.

[13] X. Zheng and X. Cao, "Research on lineal gradual forgetting collaborative filtering algorithm.," *Jisuanji Gongcheng/ Computer Engineering*, 2007.

[14] Y. Zhou, T. Wang, and X. Zhao, "A hybrid recommendation algorithm based on time factor," *International Journal of Security and Networks*, Oct. 2015.