Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Tree-based Contextual Learning for Online Job or Candidate Recommendation with Big Data Support in Professional Social Networks

WENBO CHEN[1], (Student Member, IEEE), PAN ZHOU[1], (Member, IEEE), SHAOKANG DONG[2], SHIMIN GONG[3], (Member, IEEE), MENGLAN HU[1], (Member, IEEE), KEHAO WANG[4] and DAPENG WU[5], (Fellow, IEEE)

[1]School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan, 430074, China
[2]Department of Computer Science and Technology, Nanjing University, Nanjing, 210093, China.
[3]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong, 518055, China.
[4]Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA.
[5]Department of Electrical and Computer Engineering, University of Florida, Gainesville, Florida, 32611, USA.

Corresponding author: Pan Zhou (e-mail: panzhou@hust.edu.cn).

ABSTRACT Enabling users conveniently accessible to their most interesting jobs or candidates is an important and challenging task for professional network recommenders (PNRs) in professional social networks (PSNs). PSNs nowadays accommodate tremendous and diverse users and face continuous uploading and high data freshness. In this paper, an online mining and predicting system is proposed for personalized job or candidate recommendation with big-data support. It considers the users' explicit information as context to achieve personalized recommendation. In addition, the system utilizes the reward extracted from online implicit information of the previous users with similar context information to relieve the cold start problem. Furthermore, a tree-based method is introduced to address large-volume items by effectively analyzing them in cluster level and thus reduce the computational load. Considering the dynamic property of PSNs, our model is adaptive for the expanding dataset, enabling it to real-timely make accurate recommendations for the incessant new arrivals. Moreover, theoretical analysis shows that our algorithm achieves sublinear regret over time. Finally, extensive experiments are conducted to test our algorithm based on the real dataset from ACM RecSys Challenge[a] and validate the outstanding performance of our algorithm when compared with other existing algorithms.

[a]http://www.recsyschallenge.com/2017

INDEX TERMS Big data, contextual online learning, bilateral recommendation, professional social networks.

I. INTRODUCTION

A. MOTIVATION

THE tremendous advancement of network technique promotes the prevalence of worldwide professional network services. In professional social networks (PSNs), a large number of recruiters post jobs and job seekers post their CVs, trying to find their targets (i.e., suitable candidates and desirable jobs respectively). According to the Wikipedia, the 20th most popular website Linkedin, as of April 2017, had 500 million members in 200 countries, out of which more than 106 million members are active[1]. However, users are flooded in the sea of the vast and heterogeneous information and are struggling to make decisions [1]. To facilitate the efficiency of users, it is crucial to propose an effective mining and prediction system which can recommend appropriate jobs or candidates to users.

The problem of recommending items in PSNs poses several modeling and system challenges. First, personalized professional network recommenders (PNRs) require promptly

[1]https://en.wikipedia.org/wiki/LinkedIn

make accurate recommendations to the job seekers and employers and real-timely update the system's policy to maximize users' satisfaction. To achieve personalization, user's explicit information (e.g., candidates' age, gender and experiences and jobs' type, salary and key skills) should be effectively used. However, recommendations only based on explicit information could result in high risks such as a higher number of dissatisfied job seekers and potentially longer unemployment durations [3]. Therefore, implicit information concerning users' online interactions also requires consideration. Implicit information consists of all signals about users' interests that can be deduced by their online behaviors such as the pages they visited, the time they stayed on a specific page and the items they saved for revisiting [3]. Therefore, we formulate the job or candidate recommendation as an online learning model by introducing the explicit information as context to cluster the items and extracting reward from the implicit information to update the policy, striving for the users' satisfaction maximization in the long run.

Second, job or candidate recommendation faces its unique challenges compared with other types recommendations (e.g., book, movie). The challenge is incurred by the fact that a job posting is meant to hire one or few employees only and a job seeker aims to find one or few jobs to apply [4]. Once a job recruits enough staffs or an employee finds desirable works, they might pay scant heed for the future recommendations and cannot be further recommended to new arrivals. Therefore, the active time of jobs or employees in PSN is comparatively transitory and can get expired quickly. In [5], expired items are detected and removed. Nevertheless, it is a waste of valuable historical data since the related contextual and reward information can be further utilized for future recommendations. Therefore, the recommender should be capable to efficiently solve the expired items and harness the historical data related to them.

Third, while personalization brings the system accurate recommendation when facing the great diversity of both jobs and candidates, it suffers from the high computational load. This is because it needs to real-timely score millions of items (i.e., candidates or jobs) considering the users' context. As a result, some existing recommenders [6]–[8] that score items individually cannot effectively and real-timely handle this issue. In addition, the frequent arrivals of new users render cold start a severe issue. This is because very few prior interaction histories of the diverse users are available at the very beginning. Furthermore, in the big-data era, the system should have a strong scalability for the vast amount of incoming jobs and users. Consequently, an online PNR that both considers the implicit and explicit information of users and has the capacity to real-timely and effectively tackle a large volume of data is needed for PSNs.

B. RELATED WORKS

A plethora of works focuses on jobs and candidates recommendation, which can be categorized as textual-based recommendation and filtering-based recommendation. The textual-based recommendation systems identify the semantic content by analyzing the users' provided information as well as the content of job offers. To this end, there are two kinds of semantic analysis methods: the ontology-based approaches [10] and text-mining-based approaches [11], [12]. However, most of the existing recommendations in this domain only pay attention to candidate selection by human resources rather than bilateral recommendations for the job or candidate [6]. Moreover, those works only use the explicit information proposed by users and neglect their implicit online interactions, potentially bringing about a higher number of dissatisfied job seekers [3].

As for filtering recommendations, there are three branches: content-based filtering, collaborative filtering and hybrid filtering. Content-based filtering [7], [13]–[16] relies on prior behavior or preferences of a user to recommend items [17]. Nevertheless, content-based filtering can easily bog down in the mud of overspecialization i.e., the cases where the users' recommended items are quite similar to what they have previously rated [8]. Collaborative-filtering recommender systems [18]–[21], finding the similar users and predicting the preferred item through the historical information, may be accurate when the degree of difference between users is not significant. However, collaborative-filtering methods require enough historical records and feedbacks from users, which may cause cold start issue when facing the frequent arrivals of vast and dynamic job seekers and recruiters [22]. Hybrid-filtering systems combine the abovementioned methods and can somewhat relieve the overspecialization and cold start problems [6], [8]. Nevertheless, those systems cannot well tackle the large-scale dataset because scoring tremendous items or users individually at each round is too computational cost.

In online learning, bandit algorithms get a great attention recently. Contextual multi-armed bandit algorithms (CMAB) are wildly used in recommendation system [23]–[29], which can guarantee the optimal performance in finite rounds and achieve personalized recommendation. These works assume that the reward of an item from users with similar contexts meets the Lipschitz condition, which is the same as our algorithm. The benefit of contextual bandit model is that the system can serve new users by utilizing the historical rewards from those who have the similar context information, thus greatly relieving cold problem. Nevertheless, the state-of-the-art CMAB algorithms cannot well cope with the issues in PSNs. For example, the author in [28] proposes LinUCB algorithm to solve recommendation problems of the news article, in which the payoff (reward) of a recommendation is modeled as a linear function of the context with unknown weight vector. However, the linearity assumption may not be practically satisfied in the scenario of PSNs. In [33], an item-cluster tree is proposed to achieve lower complexity, but the tree is built from the leaf to root offline. Therefore, the total number of items must be known ahead, which is impractical in the big data scenarios. Furthermore, a series of theories named $\mathcal{X}$-armed bandit [30], [32] have been proposed. The

core purpose of them is to optimize an unknown function. $\mathcal{X}$-armed bandit algorithms consider the set of arms in a generic measurable space and thus can simulate the big-data scenario. But those works cannot be easily applied in this job and candidate recommendation system due to their neglecting of users' context information, thus unable to achieve personalized recommendations.

C. CONTRIBUTION

To solve the abovementioned challenges, we propose a tree-based and context-aware online recommendation system for both job seekers and recruiters in PSNs. Our method is derived from CMAB [27]. CMAB keeps variables and each of them carries the information from the tradeoff between estimated performance and uncertainty of each arm (or action). This variable is updated at each round by learning the historical feedbacks (rewards). Then, the algorithm selects the arm with the highest variable at each round. Furthermore, in the next round, the decision-making strategies for all arms are updated based on the feedback of the selected arm. The performance loss in CMAB is evaluated in term of regret (i.e., the gap between mean reward and empirical reward). The algorithm converges to the optimal policy when it can achieve sublinear regret.

To relieve the cold start problem, we make the Lipschitz condition assumption of the reward of similar users. Therefore when new jobs or candidates come to the system, though there is no historical information about them, the system can utilize the historical rewards of their counterparts. Then the system can deduce their potentially favored items and make reasonable recommendations.

To effectively harness the massive data resource in PSNs, a scoring-tree structure is introduced to cluster the candidates and jobs and score them in cluster level to cope with the large-scale and increasing dataset. Each cluster in the structure is projected as an arm in CMAB and scored by the implicit feedbacks extracted from users' natural behaviors. Differing from the offline and bottom-up tree structure in [33], the tree structure updates online and expands from the top to the bottom, which is more appropriate for the recommendation in big-data scenario. In addition, compared with our prior work [34] in which the context partition operates at the very beginning depending on the total number of rounds, the new proposed dynamic partition method divides the context space based on the context of the new arrival, though has slightly higher computing load, leading to more accurate and personalized recommendations. Furthermore, different from tree-base methods proposed in [33], [34], in which the number of items in the tree structure is fixed after its building, our new proposed algorithm is adaptive for increasing items with the consideration that items may be expired. This paper makes the following contributions:

- We propose a contextual online learning algorithm for job or candidate recommendation applied on the PSNs.
- In order to harness the large-scale items, the proposed system cluster the similar items based on the dissimilarity function. The cluster-level operation and analysis guarantee the efficiency of the system.
- For more accurate and personalized recommendation, the system both consider the explicit and implicit information of users and propose a dynamic context partition method, which divides the context space depending on the context of the new arrival.
- Extensive and practical simulations are performed to evaluate the performance of the proposed algorithm.

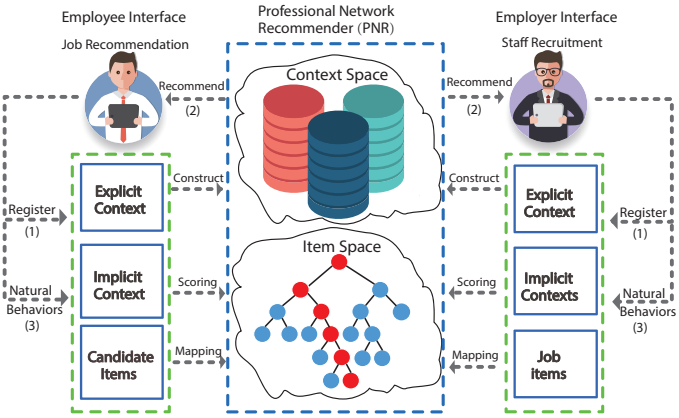


FIGURE 1: System Model

The reminder of this paper is organized as follows: In Section II, we formalize the PNR problem and present the system model. In section III, we present our algorithm ETDC, some optimization details of the crucial parameter and the analysis of ETDC's regret bound. Experimental design and simulation results are available in Section V, and the paper is concluded in Section VI.

II. PROBLEM FORMULATION

There are four major components in our proposed professional network recommendation system: employee, recruiter, job and PNR. On the one hand, employees input their personal information or update their resumes through the employee interface for their preferred jobs. In this case, PNR recommends jobs to employees. On the other hand, recruiters post the job positions with various requirements for suitable workers. In this case, PNR recommends employees to recruiters. Since the recommended items and corresponding users differ in the scenarios of job recommendation and employee recruitment, we use item and user in the following parts to simplify the discussion. We first briefly introduce the workflow of our system. We consider a time-slotted system in this paper. As shown in Fig. 1, at each time slot $t = 1, 2, 3, ...$, the system works in the following steps: 1) A user arrives PSN and PNR gets the context vector extracted from the user's explicit information; 2) the system observes the context and recommends an item to the user based on the current context as well as the historical information about users, items, contexts and rewards; 3) after the recommendation, PNR extracts reward from the user's natural click behaviors.

Then PNR adds the interaction log to the database which will be used in future recommendation. Then we detailedly illustrate the components:

User: In PSNs, users are either employees or recruiters. We denote by $\mathcal{U} = \{u_1, u_2, ...\}$ the users set and $\mathcal{C} = \{c_1, c_2, ...\}$ their corresponding contexts. The context vector is drawn from a d_C-dimensional context space $\mathcal{C}$, of which each dimension maps one feature of the user. For instance, if a recruiter u poses a job with background information (e.g., geographic location, job position and salary, inclination for employees expert in Java, machine learning and algorithm design). Then the PNR extracts the context vector c from the abovementioned explicit information. For simplicity of exposition, we normalize the context space into $\mathcal{C} = [0, 1]^{d_C}$. In the context space, we define D_c as a metric $D_c : \mathcal{C} \times \mathcal{C} \to \mathbb{R}+$. A larger $D_c(c, c')$ shows that the two contexts c and c' exhibit higher dissimilarity. The metric D_c can be the Euclidean norm or any other norm in the Euclidean space $\mathcal{C}$.

Item: Items are either the jobs in job recommendation or the employees in employee recruitment. Specifically, we denote by $\mathcal{X} = \{x_1, x_2, ...\}$ the total corresponding items, where each x represents the item feature vector, to construct a d_I-dimensional space. In the item space, we define the dissimilarity function $D_i : \mathcal{X} \times \mathcal{X} \to \mathbb{R}+$. Similar to D_c, a larger $D_i(x, x')$ indicates that the two item x and x' exhibit higher dissimilarity. As mentioned in the introdution, the items in PNR are dynamic i.e, the recommended item set varies with old items expiring and new items arriving.

In order to support the big-data analysis, an infinite item-scoring tree structure is built to contain item space $\mathcal{X}$, which enables the analysis in cluster level and thus reduces the scale of analyzed items at each round. Let $\mathcal{T}$ denote a binary tree, of which each node covers a subset of $\mathcal{X}$ and the two children nodes are mutually exclusive and combine into their father node. We denote by (h, i) the i^{th} node at depth h in $\mathcal{T}$, where the index i of nodes at depth h is restricted by $1 \leq i \leq 2^h$. The set of total items in the node (h, i) is denoted by $\mathcal{P}_{h,i}$. Thus, we obtain: $\mathcal{P}_{0,1} = \mathcal{X}, \mathcal{P}_{h,i} = \mathcal{P}_{h+1,2i-1} \cup \mathcal{P}_{h+1,2i}$, and $\mathcal{P}_{h+1,2i-1} \cap \mathcal{P}_{h+1,2i} = \varnothing$, where $\varnothing$ denotes empty set. For instance, root node $(0, 1)$ covers the whole item space $\mathcal{X}$ and its child nodes $(1, 1)$ and $(1, 2)$ cover the two mutually exclusive subspaces of $\mathcal{X}$. In the binary tree, we define the diameter of the node (h, i) as $diam((h, i)) := sup_{x,y \in \mathcal{P}_{(h,i)}} D_i(x, y)$, and an item ball with radius $\epsilon > 0$ and center item $x \in \mathcal{X}$ is defined as $\mathcal{B}(x, \epsilon) := \{x' \in \mathcal{X} : D_i(x, x') \leq \epsilon\}$.

At certain timing, our tree will grow and expand two child nodes to enable more accurate recommendations in future (detailed in Section III). The growth of tree is shown in Fig. 2. Therefore it is intuitive that our tree should expand following some geometric rules i.e., the size of its node should decrease exponentially with its depth increasing. We formalize our intuition with the following assumption:

Assumption 1 (Well-shape growth). There exists constants $v_1, v_2 > 0$ and $0 < \rho < 1$ such that for all nodes (h, i):

(a) $diam((h, i)) \leq v_1 \rho^h$,

(b) $\exists x^o_{h,i} \in \mathcal{P}_{h,i}$ such that $\mathcal{B}_{h,i} := \mathcal{B}(x^o_{h,i}, v_2 \rho^h) \subset \mathcal{P}_{h,i}$.

PNR: The PNR works both for job recommendation and employee recruitment. For different scenarios, PNR learns online the performance of corresponding users based on all the historical responses and then recommends the appropriate item for each arriving users to maximize the cumulative reward of them. We denote by r the rewards from users, which are extracted from users' implicit information (i.e., online interactions). PNR holds the database containing the historical interaction logs between users and items in the form of triple (c, x, r). We denote by $\mathcal{M} = \{(c_1, x_1, r_1), (c_2, x_2, r_2), ...\}$ the set of logs.

As mentioned in the Section I-A, the reward is determined by the users' implicit information i.e., natural behaviors after accepting the recommendation. Given the less robust nature of the implicit information, it is possible that binary representation of user interest (either 1 or 0) will lead to a better recommender system [38]. Therefore, We use the interest indicators in existing recommendation systems [35]–[37] to model the binary reward:

$$r_{x,c} = li(x, c) \cdot \mathbb{I}\{re(x, c) > 0 \; or \; sa(x, c) = 1\}, \quad (1)$$

where $re(x, c)$ is the reading time that the user with context c spent on the recommended item x, which is measured as the difference in seconds between the HTTP-request to view the recommended item's web page and the subsequent HTTP-request [38]; $sa(x, c)$ indicates whether the user save the item in his/her profile e.g., when $sa(x, c) = 1$, the user with context c saved the item x after the recommendation; $li(x, c) \in \{1, 0\}$ shows whether the user clicked a button that indicates the item is not to his/her liking e.g., $li(x, c) = 0$ illustrates that the user with context c clicked the "unlike" or "delete" button for the item x. Therefore, we can see, at each time slot, the r is a Bernoulli distribution which takes value 1 with unknown probability determined by context c and item x. Similar with [23], [24], [33], the reward is assumed to be an identically and independently distributed (i.i.d.) random variable. Then, we define the expected reward function of an item x for a context c as $f(x, c) = \mathbb{E}[r_{x,c}]$. For a given context c, we denote the optimal item as $x^*_c = \arg\max f(x, c), \forall x \in \mathcal{X}$.

In previous recommendation system based on contextual bandit algorithms, Lipschitz condition is used to model the similar expected rewards of an item in two similar contexts [28], [33]. Similar to those works, we formalize this dependency as follows:

Assumption 2 (Local Smoothness). Given $c \in \mathcal{C}$, for all $x \in \mathcal{X}$:

$$f(x^*, c) - f(x, c) \leq D_i(x^*, x).$$

Different from the works [28], [33] which require the smoothness information around all items, we weaken the assumption and only the smoothness information around the optimal item x^* is needed. Based on the assumption, we bound the gap between the expected reward of the optimal

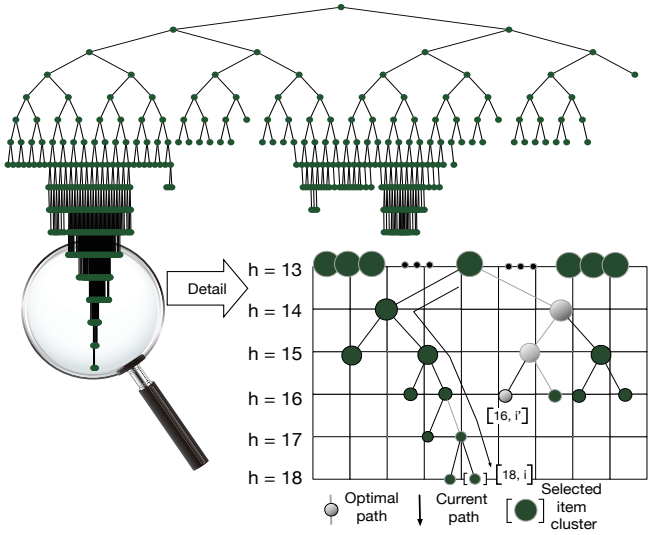


FIGURE 2: Item Tree Structure

item and other items. Furthermore, the assumption indicates the intuition that compared with the optimal item, the more similar the item we recommend, the larger probability the user has a positive feedback about it. For instance, in job recommendation, if an employee wants to be an algorithm designer, she is more likely to click the "save" button for a programming-related job than a commercial work. Then, we similarly formalize the relation between rewards and contexts.

Assumption 3 (Lipschitz Condition). For the same item x, given two context vectors c_1 and c_2 there exists a constant L_C such that:

$$\|f(x, c_1) - f(x, c_2)\| \leq L_c D_c(c_1, c_2).$$

Apparently, the similarity of contexts of users determines the similarity of feedback with regard to the same item. For instance, the natural click behaviors of two employees after accepting the recommendation of the same job are determined by their interests towards it and their interests are closely related to their context information.

At each time slot t, a natural gap exists between the real reward r_t and the optimally expected reward $f(x_t^*, c_t)$. This gap is mainly caused by the insufficient learning of our algorithm. For example, in Fig. 2, we take a portion of the tree to illustrate the gap. At this round, we search the tree structure and recommend the item in the node $(18, i)$. However, the actual optimal item x_t^* is in $(16, i')$. Therefore, there is a gap between the real reward from recommending the item in the node $(18, i)$ and the expected optimal reward from recommending the node $(16, i')$. This deviation comes from the system's not converging to the best policy. Therefore, in order to measure the reward loss, we introduce regret as the metric.

Definition 1 (The Regret of Online Learning). After the time n, the cumulative gap between the optimal items and selected items are:

$$R(n) = \sum_{t=1}^{n} f(x_t^*, c_t) - \sum_{t=1}^{n} r_t, \tag{2}$$

the cumulative regret also represents the convergence rate of the total average reward of the learning algorithm and in the following subsections we will both theoretically(section III) and experimentally (section IV) prove that the regret of our algorithm achieves sublinear over time.

III. PROPOSED ALGORITHM

In this section, we present our Expandingly Tree-based and Dynamically Context-aware Online Learning algorithm (ETDC) and analyze its regret through time T. The main algorithm is shown in Algorithm 1, and its subfunctions are illustrated in Algorithm 2 and 3.

A. ALGORITHM DESCRIPTION

For better clarification, the algorithm is discussed in the scenario of job recommendation, where users are employees and items are jobs and jobs are recommended to arriving users. In this scenario, ETDC runs in four steps. 1) ETDC first checks the incoming jobs and add the new jobs into the tree structure at each round since new jobs can be posted by the companies at any time, 2) Then, with an employee arriving at the platform, the algorithm predicts the employee's preference by aggregating historical rewards from the employees whose context vectors are similar to the arrival. 3) With the rewards, ETDC updates the scoring-tree structure and then search for the most promising job and recommends it to the employee. 4) After that, the algorithm extracts the reward from the employees' browsing behavior and updates the log to facilitate future recommendation. Note that our algorithm is also compatible with employee recruitment. Then we provide the detailed descriptions of each step:

1) Adaptive for the Incoming Items: At each time slot t, ETDC first check whether new jobs are posted by the companies (line 2-4 in ETDC). If a new job arrives the system, ETDC extracts its feature vector and call the subfunction AddItem to add the job into the tree structure. Specifically, since the root node contains the whole job space, AddItem first adds the job vector into it (line 1 in AddItem). Then, AddItem trys to add the job vector into its two children nodes. Note that the job space boundary of the node is defined by the maximum distance between any two job vectors in the job space. Therefore, the space mapped by the nodes may expand (measured by the increase of the diameter) due to adding new job vector into it. On this basis, our goal is to minimize the increase of diameter of each child nodes to reduce the overlapping of their corresponding feature spaces (line 3 in AddItem). Finally, the algorithm adds the job into the node which has a smaller increase of the diameter at each depth and recursively repeats the process till reaches the leaf node (line 3-8 in AddItem). The Fig. 3 illustrates how ETDC adds the new item at depth $h + 1$. We can see that the increase of diameter of $(h + 1, 2i)$ after adding a new item is smaller than that of $(h + 1, 2i - 1)$. Therefore, the incoming item is

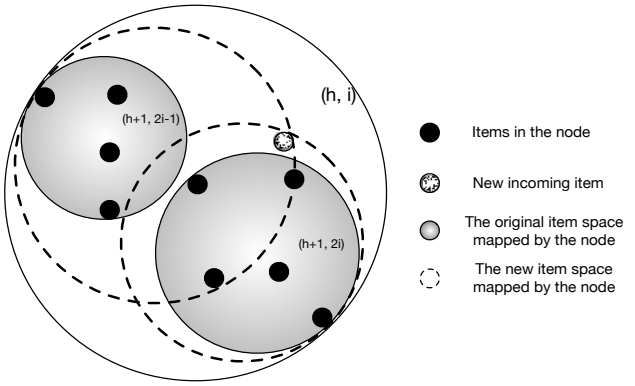


FIGURE 3: Adaptive for the Incoming Items

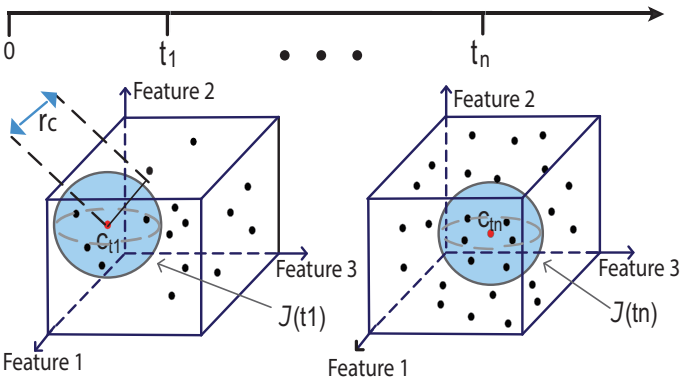


FIGURE 4: Finding the Similar Contexts

added into the node $(h+1, 2i)$. This process guarantees that the space overlapping is minimized between the nodes in the same depth, which is benefit to define a tighter upper bound of the expected reward of each node.

2) Obtain the Historical Rewards: At each time slot t, ETDC receives an employee's context c_t extracted from the explicit information of u_t. Then it searches for the employees who have similar context with c_t in the context space. The similarity between two employees is defined as Euclidean distance. We assume that there exists a threshold r_c and two employees are similar when their distance is less then r_c. We can see that r_c should be carefully selected and is related with the total number of rounds T, which will be detailed in the theoretical part (Section III-B). Therefore, even when a new employee arrives, the algorithm can still use the historical rewards from those similar employees to predict the new arrival's preference. As a result, the cold start problem can be mitigated. As illustrated in Fig. 4, at different time slot, ETDC finds the similar employees in the context space based on the new arriving context vector. The set of the context of those similar employees is defined as $\mathcal{J}(t) = \{c_\gamma : \gamma < t,\ D_c(c_\gamma, c_t) \leq r_c\}$. Thus, we denote the set that contains the arriving time slot of past users in the $\mathcal{J}(t)$ by: $\Gamma(t) = \{\gamma : \gamma < t, c_\gamma \in \mathcal{J}(t)\}$. As is mentioned before, PNR holds the database containing the historical interaction logs in the form of triple (c_t, x_t, r_t). Therefore, we can get a set of tuples: $\mathcal{R}(t) = \{(x_\gamma, r_\gamma) : \gamma \in \Gamma(t)\}$, which will be used to score the jobs in the following phase (line 6 in ETDC).

3) Score the Item Tree: In order to guarantee the personalized recommendation, ETDC predicts the employee's preference by updating the scores of each job nodes in the whole binary tree based on $\mathcal{R}(t)$ obtained in phase 2. Intuitively, a higher score indicates the employees tends to have a stronger inclination to the jobs. We first define the number of times that the node (h, i) has been explored up to time t:

$$T_{h,i}(t) = \sum_{(x_\gamma, r_\gamma) \in \mathcal{R}(t)} \mathbb{III}\{x_\gamma \in (h, i)\}. \quad (3)$$

In order to estimate the expected reward function $f(x)$, we record the empirical reward in every node, so the estimation

Algorithm 1 ETDC

Require: The constants $v_1, c > 0, r_c, T, d_C, d_I, \rho \in (0, 1)$
Initialize: $t = 1, j = 1, \mathcal{T}_t = \{(0, 1), (1, 1), (1, 2)\}, E_{1,1}(t) = E_{1,2}(t) = \infty, \mathcal{M} = \{\}, H = 1, h = 0.$

1: **for** $t = 1, 2, ...T$ **do**
2: **if** a new item x is added **then**
3: $\mathcal{T}_t \leftarrow \mathsf{AddItem}(\mathcal{T}_t, x)$
4: **end if**
5: Extract the context vector c_t from the user u_t
6: Get similar context set $\mathcal{J}(t)$, $\Gamma(t)$ and $\mathcal{R}(t)$
7: **for** all nodes $(h, i) \in \mathcal{T}_t$ **do**
8: Update the value $T_{h,i}(t)$ in Eq.(4);
9: Update the value $E_{h,i}(t)$ in Eq.(6);
10: Update the value $S_{h,i}(t)$ in Eq.(5);
11: **end for**
12: $(h_t, i_t) \leftarrow \mathsf{OptimalPath}(\mathcal{T}_t)$
13: Randomly select an unexpired item in node $\mathcal{P}_{h_t, i_t}$
14: Calculate the the reward r_t in Eq.(1)
15: $\mathcal{M} \leftarrow \mathcal{M} \bigcup \{(c_t, x_t, r_t)\}$
16: Calculate the threshold value $\tau_h(t) \leftarrow \frac{c^2 \ln(t)}{v_1^2} \rho^{-2h_t}$
17: **if** $T_{h,i}(t) \geq \tau_{h_t}(t)$ **and** $(h_t, i_t) \in Leaf(\mathcal{T}_t)$ **then**
18: Expand the current node:
$\mathcal{T}_t \leftarrow \mathcal{T}_t \cup \{(h_t + 1, 2i_t - 1), (h_t + 1, 2i_t)\}$
19: $H \leftarrow H + 1$
20: **end if**
21: **end for**

of empirical reward is denoted by:

$$\hat{\mu}_{h,i} = (1/T_{h,i}) \sum_{(x_\gamma, r_\gamma) \in \mathcal{R}(t)} r_\gamma \mathbb{III}\{x_\gamma \in (h, i)\}. \quad (4)$$

Then we introduce scoring function $S_{h,i}(t)$ of each node i.e., a tight upper bound of the expected reward of each job node:

$$S_{h,i}(t) = \begin{cases} E_{h,i}(t) & (h, i) \in Leaf(\mathcal{T}_t) \\ \infty & T_{h,i}(t) = 0 \\ \min\{E_{h,i}(t), \max\limits_{k \in \{2i-2, 2i\}} S_{h+1,k}(t)\} & otherwise \end{cases}, \quad (5)$$

where $E_{h,i}(t)$ denotes the upper-bound for the ideal reward of the node (h, i) and it is defined as:

$$E_{h,i}(t) = \hat{\mu}_{h,i}(t) + \sqrt{\frac{c^2 \ln(t)}{T_{h,i}(t)}} + v_1 \rho^h + L_c r_c. \quad (6)$$

The intuition of the scoring function comes from the tradeoff between the exploitation and exploration. The first term $\hat{\mu}_{h,i}(t)$ indicates the exploitation. In other words, the employee u_t are more likely to give a high reward to those jobs whose historical rewards are high from the employees with similar context to her. The second term $\sqrt{\frac{c^2 \ln(t)}{T_{h,i}(t)}}$ indicates the exploration. It considers the fact that employee's attitude

Algorithm 2 AddItem

Require: Tree $\mathcal{T}$, x
Initialize: $(h,i) = (0,1)$
1: $\mathcal{P}_{h,i} \leftarrow \mathcal{P}_{h,i} \cup x$
2: **while** $(h,i) \notin Leaf(\mathcal{T})$ **do**
3: **if** $diam((h+1,2i-1) \cup x) - diam((h+1,2i-1))$ $\leq diam((h+1,2i) \cup x) - diam((h+1,2i))$ **then**
4: $(h,i) \leftarrow (h+1,2i-1)$
5: $\mathcal{P}_{h,i} \leftarrow \mathcal{P}_{h,i} \cup x$
6: **else**
7: $(h,i) \leftarrow (h+1,2i)$
8: $\mathcal{P}_{h,i} \leftarrow \mathcal{P}_{h,i} \cup x$
9: **end if**
10: **end while**
11: **return** $\mathcal{T}$

Algorithm 3 OptimalPath

Require: Tree $\mathcal{T}$
Initialize: $(h,i) = (0,1), T_{0,1} = \tau_0(t) = 1$
1: **while** $(h,i) \notin Leaf(\mathcal{T}_j)$ **and** $T_{h,i}(t) \geq \tau_h(t)$ **do**
2: **if** $S_{h+1,2i-1} \geq S_{h+1,2i}$ **then**
3: $(h,i) \leftarrow (h+1,2i-1)$
4: **else**
5: $(h,i) \leftarrow (h+1,2i)$
6: **end if**
7: **end while**
8: **return** (h,i)

towards the same job may vary with time. For example, an employee who holds a negative attitude for a job at first might be willing to accept the job in future. Therefore, the scoring function also considers those job clusters that have seldom been selected. The third term $v_1\rho^h$ is to bound the deviation in a node, which is based on the Assumption 1 and the fourth term $L_c r_c$ is to bound the gap caused by the partition of context space, which is derived from Assumption 3. Since the scoring function $S_{h,i}(t)$ of node (h,j) is related to its child nodes, ETDC postorderly traverse and score each node at each time slot in Eq. 5 (line 7-11 in ETDC).

4) Explore the Tree and Expand the Leaf Node: In this phase, ETDC calls the subfunction OptimalPath to explore the tree and find the job node with the highest score (line 12 in ETDC). In the binary tree, we always choose the children node with higher score till the current node is a leaf (line 2-6 in OptimalPath). After that, ETDC randomly recommends an unexpired job x_{h_t,i_t} in the leaf node (h_t,i_t). Then ETDC observes the employee's online behavior and calculates the reward r_t based on Eq.1. The triple (c_t,x_t,r_t) is then stored in the database(line 13-15 in ETDC).

Then, ETDC decides whether to expand the leaf nodes to refine the tree structure. An intuitive way is to expand the selected leaf nodes at each round. However, this may lead to more errors in stochastic situations especially in the scenario of PNR where the rewards are extracted from users' online browsing behaviors which have a less robust nature [38]. Therefore, it is important to determine the timing to expand the leaf node (line 16-20 in ETDC). We define the threshold $\tau_h(t) = c^2\ln(t)\rho^{-2h_t}/v_1^2$. When the explored times $T_{h_t,i_t}(t)$ of node (h_t,i_t) reach the threshold, ETDC adds the node two children nodes. This threshold is derived from two time-varying terms in $E_{h,i}(t)$. The first term $v_1\rho^{h_t}$ bounds the gap of all items in the same node, which is dependent on the depth of a tree. The second term $\sqrt{c^2\ln(t)/T_{h_t,i_t}(t)}$ decreases with the number of exploited times increasing. When $\sqrt{c^2\ln(t)/T_{h_t,i_t}(t)}$ is smaller than $v_1\rho^{h_t}$, we ensure that the uncertainty of a node is more related to the systematic gap of the items in the node than the exploited times. Thus the node (h_t,i_t) is fully exploited and its errors incurred by less robust nature of those rewards drop into a tolerant level. Therefore, we expand the leaf node to increase the depth of the tree when the term $v_1\rho^{h_t}$ is equal to the term $\sqrt{c^2\ln(t)/\tau_h(t)}$. As a result, we derive the threshold: $\tau_h(t) = c^2\ln(t)\rho^{-2h_t}/v_1^2$.

B. ALGORITHM ANALYSIS

We prove that the regret of ETDC has sublinear convergency rate over time in this subsection. We first define the ϵ-optimal item set as $\mathcal{X}_\epsilon = \{x \in \mathcal{X} : f(x^*,c_t) - f(x,c_t) \leq \epsilon\}$, where $\epsilon \geq 0$. Therefore, with the help of ϵ-optimal item set, we can characterize the scale of the problem by introducing packing number. For $\epsilon' \leq \epsilon$, there is a constant C_I such that $\mathcal{N}(\mathcal{X}_\epsilon, D_i, \epsilon') \leq C_I(\epsilon')^{-d_I}$, where $\mathcal{N}(\mathcal{X}_\epsilon, D_i, \epsilon')$ denotes the maximum number of disjoint suprapheres with radius ϵ' in the space $\mathcal{X}_\epsilon$ w.r.t the metric D_i. Similarly, in the context space $\mathcal{C}$, we have the packing constant C_c such that $\mathcal{N}(\mathcal{C}, D_c, r_c) \leq C_c(r_c)^{-d_C}$, where $\mathcal{N}(\mathcal{C}, D_c, r_c)$ denote the minimal number of suprasphere with radius $r_c \in (0,1)$ that can cover the context space $\mathcal{C}$ w.r.t the metric D_c.

In the algorithm, a threshold $\tau_h(t)$ is introduced to determin the expansion of leaf node. On the one hand, the threshold ensures the accuracy of the empirical reward. On the other hand, it can help us bound the maximum depth of the cover tree. Thus, we present a lemma about the explored depth of the tree over time t.

Lemma 1. *Given the threshold value $\tau_h(t)$, the depth of every tree $H(t)$ can be bounded as*

$$H(t) \leq \frac{1}{2(1-\rho)}\log_2(\frac{{v_1}^2 t}{c^2\rho^2}). \tag{7}$$

Proof. In the binary tree, the set of all nodes at depth h up to time t is denoted by $I_h(t)$ and the subsets of $I_h(t)$ which only contains the internal nodes or only contains the leaf nodes at depth h are denoted by $I_h^{\bar{L}}(t)$ and $I_h^L(t)$. Thus we can get

$$\begin{aligned} t &= \sum_{h=0}^{H(t)} \sum_{i\in I_h(t)} T_{h,i}(t) \geq \sum_{h=0}^{H(t)-1} \sum_{i\in I_h(t)} T_{h,i}(t) \\ &\geq \sum_{h=0}^{H(t)-1} \sum_{i\in I_h^{\bar{L}}(t)} T_{h,i}(t) \geq \sum_{h=0}^{H(t)-1} \sum_{i\in I_h^{\bar{L}}(t)} \tau_{h,i}(t) \\ &\geq \sum_{h=0}^{H(t)-1} \frac{c^2}{v_1^2}\rho^{-2h} \geq \frac{(c\rho)^2}{v_1^2}\rho^{-2H(t)}. \end{aligned}$$

As the inequality shows, even if the extreme situation that all the users are in the same sphere up to time t happens, the

depth $H(t)$ also meets the bound. Then we can obtain

$$\rho^{-2H(t)} \leq \frac{v_1^2 t}{(c\rho)^2}.$$
$$\Rightarrow H(t) \leq \frac{1}{2}\log_2 \frac{v_1^2 t}{(c\rho)^2} / \log_2(\frac{1}{\rho}) \leq \frac{1}{2(1-\rho)}\log_2(\frac{{v_1}^2 t}{c^2\rho^2}).$$

□

Lemma 1 shows that the expansion rate of the cover tree is no more than $\mathcal{O}(\ln n)$. Then, we introduce an event:

$$\eta_t = \{\forall 0 \leq T_{h,i}(t) \leq t, \forall h \geq 0, \forall 0 \leq i \leq 2^h : |\hat{\mu}_{h,i}(t) - f(x_{h,i})| \leq c\sqrt{\ln(t)/T_{h,i}(t)} + L_c r_c\}. \quad (8)$$

If this event holds, it is easy to bound $\hat{\mu}_{h,i}$ through the expected reward and the gap. Then, in Lemma 2, we present that this event holds with a high probability.

Lemma 2. *If $c = 3\sqrt{(1-\rho)/2}$, for any value t the event η_t holds with probability at least $1 - 1/t^6$.*

Proof. The proof of the Lemma 2 is available in the Appendix A of [40]. □

Note that the value of constant c is changeable and it is relevant to the probability to hold the event η_t. We can enlarge the value of c to further ensure the happen of η_t, but the enlarging of c hinders the expansion of the tree (from Lemma 1), which reduces the efficiency of the learning process. Finally, we choose a trade-off value $c = 3\sqrt{(1-\rho)/2}$. Then, we define the instantaneous regret as $\delta(t) = f^*(t) - r_t$ and decompose the regret as two terms under the events η_t and η_t^c, where event η_t^c is the complementary event of η_t. Then the cumulative regret can be expressed as

$$R(n) = \sum_{t=1}^{n} \delta(t) = \sum_{t=1}^{n} \delta(t) I(\eta_t) + \sum_{t=1}^{n} \delta(t) I(\eta_t^c) = R^{\eta}(n) + R^{\eta^c}(n). \quad (9)$$

Since our goal is to calculate the cumulative regret $R(n)$, therefore we shall calculate $R^{\eta}(n)$ and R^{η^c} respectively. We first calculate the regret $R^{\eta^c}(n)$ under the event η_t^c.

Lemma 3. *Given the constant $c = 3\sqrt{(1-\rho)/2}$, the regret of low-probability term $R^{\eta^c}(n)$ is bounded as $R^{\eta^c}(n) \leq \sqrt{n \ln n}$ with probability $1 - 1/n^2$.*

Proof. The proof of the Lemma 3 is available in the Appendix A of [40]. □

The proof is in accordance with the actual situation in terms of the time. When our algorithm initially operates, there can be a large gap between the empirical reward and the expected reward, which makes the event η_t^c unlikely to holds at first. However, from the proof, we can obtain that the high-probability event η_t almost always holds after the time $\sqrt{n \ln n}$. This also indicates the good convergence of the cumulative regret through time.

Then we are to calculate the regret $R^{\eta}(n)$ under the event η_t. We denote the node chosen at time t by (h_t, i_t), the path from the root to the selected node by P_t and the parent node of (h_t, i_t) by (h_t^p, i_t^p). After that, we decompose the regret $R^{\eta}(n)$ in terms of the node space. We define the instantaneous systematic regret $\delta(h_t, i_t)$ as $f^* - f(x_{h_t,i_t})$ and the instantaneous floating regret $\delta(r_t)$ as $f(x_{h_t,i_t}) - r_t$. Therefore, the regret $R^{\eta}(n)$ can be expressed as

$$\begin{aligned} R^{\eta}(n) &= \sum_{t=1}^{n} \delta(t) I(\eta_t) \\ &= \sum_{t=1}^{n} (f^* - f(x_{h_t,i_t}) + f(x_{h_t,i_t}) - r_t)\mathbb{III}(\eta_t) \\ &= \sum_{t=1}^{n} \delta(h_t, i_t)\mathbb{III}(\eta_t) + \sum_{t=1}^{n} \delta(r_t)\mathbb{III}(\eta_t) \\ &\leq \sum_{t=1}^{n} \delta(h_t, i_t)\mathbb{III}(\eta_t) + \sum_{t=1}^{n} \delta(r_t) \\ &= R^{\eta_1}(n) + R^{\eta_2}(n), \end{aligned} \quad (10)$$

where $R^{\eta_1}(n)$ denotes the cumulative systematic regret and $R^{\eta_2}(n)$ denotes the cumulative floating regret under the event η_t. At first, we present the bound of $R^{\eta_2}(n)$.

Lemma 4. *The floating regret $R^{\eta_2}(n)$ can be bounded as $R^{\eta_2}(n) \leq 2\sqrt{n \ln n}$ with probability $1 - 1/n^2$.*

Proof. We construct the sequence $X_n = \sum_{t=1}^{n} \delta(r_t)$ ($n \geq 1$, $X_0 = 0$). By using the Azuma's inequality, we suppose that the sequence X_n is a martingale, from which we can obtain that $|X_k - X_{k-1}| = \delta(r_k) < 1$. Therefore, we have that $P[X_n - X_0 \geq 2\sqrt{n \ln n}] \leq \exp(-(2\sqrt{n \ln n})^2/2n) = \exp(-2\ln n) = 1/n^2$. So the floating regret $R^{\eta_2}(n) \leq 2\sqrt{n \ln n}$ with probability $1 - 1/n^2$. □

In order to proceed with the systematic regret $R^{\eta_1}(n)$, we must figure out some important properties for the scoring function S. Let (h_t^1, i_t^1) be a random node in the path P_t and (h_t^2, i_t^2) be the child node of (h_t^1, i_t^1). Since the root node $(0, 1)$ covers all the items, there exists at least one node (h^*, i^*) in the path P_t that contains the optimal item x^* and has the property $h^* \leq h_t^p < h_t$.

Lemma 5. *In the path P_t, for any optimal node (h^*, i^*) under the event η_t, scoring function $S_{h^*,i^*}(t)$ is always a upper bound of the value f^*.*

Proof. The proof of the Lemma 5 is available in the Appendix A of [40]. □

In Lemma 5, we prove that $S_{h^*,i^*}(t)$ is the upper bound of the optimal item's expected reward f^*. Therefore, the empirical reward $\hat{\mu}_{h,i}(t)$ floats in a certain range of the optimal reward f^*, which is in accordance with the actual situation. Then we bound cumulative regret $R(n)$.

Theorem 1. *If $c = 3\sqrt{(1-\rho)/2}$, the cumulative regret $R(n)$ of ETDC algorithm is bounded as $R(n) \leq 6L_c[\frac{9(3\sqrt{2}v_1)^{-d_C} C_c C_I {v_2}^{-d_I}}{2\sqrt{2}v_1^2(1-\rho)^2 L_c^{-d_C}}]^{\frac{1}{d_C+d_I+2}} (\ln n)^{\frac{1}{d_C+d_I+2}} n^{\frac{d_C+d_I+1}{d_C+d_I+2}} + 3\sqrt{n \ln n}$ with probability $1 - 1/n^2$.*

Proof. The proof of the Theorem 1 is available in the Appendix B of [40]. □

We have proved that the cumulative regret of our algorithm is sublinear $\mathcal{O}(n^{\frac{d_I+d_C+1}{d_I+d_C+2}} (\ln n)^{\frac{1}{d_I+d_C+2}})$, which can ensure the convergency of the reward because $\lim_{n\to+\infty} R(n)/n = 0$. The sublinear cumulative regret assures that the optimal

item can be selected with a high probability. Then we discuss the parameters related to the algorithm and their impacts based on the theoretical analyses.

- c: The value of c determines the trade off between the recommendation accuracy and the convergency rate. The increase of value of c will lead to a high convergency rate but result in the decrement of recommendation accuracy.
- d_I and d_C: As shown in the cumulative regret $\mathcal{O}(n^{\frac{d_I+d_C+1}{d_I+d_C+2}}(\ln n)^{\frac{1}{d_I+d_C+2}})$, we can obtain that if the dimension of item or context are very large, the sublinear regret will degenerate to the linear regret. In this condition, the performance of ETDC is bound to get worse, because the large dimension results in the significant sparsity in the item and context space and further highlight the "cold start" issue. On contrast, if the dimensions of item or context are too few, the relation built between the features and reward are no longer representative. In other word, the Assumption 2 and Assumption 3 may no longer hold, which will degenerate the model from context-aware to context-free. Therefore, the selection of d_I and d_C should based on the real dataset.
- r_c: In the process of proof, we optimize the value $r_c = \mathcal{O}((\ln n/n)^{\frac{1}{d_C+d_I+2}}))$ to obtain the smallest regret. The radius of the context sphere is a significant parameter for ETDC. When the radius is too large, the partition granularity of context space is coarse, which results in a large regret. However, if the radius is too small, the amount of data in each sphere will be insufficient, which is also difficult to support the accurate recommendation. We have proved that the optimal value of r_c is dependent on the execution times n. If the execution times is unknown, we can implement the doubling trick to enable ETDC still feasible for unknown time horizon [39].

IV. NUMERICAL RESULTS

In this section, we test the performance of our algorithm theoretically and experimentally. At first, we compare the regret, time complexity and space complexity with other state-of-art algorithms. Then we illustrate the dataset of our algorithm. Finally, we show the results and analyze the performance of our algorithm in various aspects.

A. REGRET, TIME COMPLEXITY AND SPACE COMPLEXITY

We first analyze the the computational cost of ETDC. Its computational cost consists of three parts: partitioning the context suprasphere, updating the cover tree and exploring the tree for the optimal item. In the first part, ETDC requires to traverse the histories and its computational cost is $\mathcal{O}(n)$. From the Lemma 1, we know that the maximum depth of the tree is no more than $\mathcal{O}(\ln n)$ up to time n. Therefore, we know that the number of nodes needed to be updated is no more than $\mathcal{O}(\ln n)$. In addition, due to the bound of the depth of the tree, the times of exploring the tree to select the node is $\mathcal{O}(\ln n)$. Thus, the total time complexity is at most $\mathcal{O}(T^2 + 2T \ln T) = \mathcal{O}(T^2)$.

We compare the regret, time complexity and space complexity with ACR algorithm [33], HCT algorithm [32] and ITFACP [34]. The specific result is shown in Table. I. In ACR, K_l is the number of nodes at depth l and the epoch E is defined as $E := \min\{L, \lfloor \log_2 T \rfloor\}$. For HCT, since it does not consider the context information, the dimension d has the similar meaning as d_I in our algorithm and it does not contain the d_C term. In ITFACP, since the partition method is static, it has smaller time and space complexity compared with our proposed algorithm when the number of context features d_C is small. However, in practice, d_C can be comparatively high to guarantee the accuracy and personalization recommendation in PSNs. With the increasing of d_C, the time and space complexity of ITFACP will degenerate greatly and converge to the same order as ETDC. In this case, the dynamically contextual partition of ETDC ensures a better performance. Since HCT is context-free, it has the lowest time and complexity among five algorithms. Nevertheless, it is also the context-free that results in the greatest deviation of its recommendations (i.e. high cumulative regret). Compared with ACR, our algorithm performs slightly better with regard to space complexity and time complexity.

B. DESCRIPTION OF THE DATABASE

The database that we use to evaluate the performance of our algorithm is from the 2017 ACM Conference on Recommender Systems challenge, which is provided by XING[2], a top leading PSN. The dataset consists of three parts: the context of users who appear on the PSN (users.csv), background information of items that should be recommended to users (items.csv) and the interaction logs between them (iteraction.csv). There are 1,497,020 users' contexts in the users.csv, of which each contains 13 features such as career_level, discipline_id, country and etc. In items.csv, there are 1,306,054 items' information, each containing 12 features such as industry_id, discipline_id and etc. On the other hand, there are more than 300 million instances in the iteraction.csv which documents the online interaction between users and items. We formulate the interaction logs as three-tuple (x, c, r), where (x, c, r) is mapped by the recommended items, context information and reward. The detailed information of the dataset is shown in Table. II.

C. BENCHMARKS

Since our algorithm belongs to the class of online learning, we compare our algorithm performance against the related approaches:

- UCB [31]: This is a classical multi-armed bandit algorithm without considering the context information.

[2]http://www.xing.com/

TABLE 1: Theoretical Comparison

Algorithm	Context-aware	Regret	Time Complexity	Space Complexity
ACR [33]	Yes	$\mathcal{O}\left(T^{\frac{d_I+d_C+1}{d_I+d_C+2}}\ln T\right)$	$\mathcal{O}\left(T^2+K_E T\right)$	$\mathcal{O}\left(\sum_{l=0}^{E} K_l+T\right)$
HCT [32]	No	$\mathcal{O}\left(T^{\frac{d+1}{d+2}}(\ln T)^{\frac{1}{d+2}}\right)$	$\mathcal{O}(T\ln T)$	$\mathcal{O}\left(T^{\frac{d}{d+2}}(\ln T)^{\frac{2}{d+2}}\right)$
ITFACP [34]	Yes	$\mathcal{O}\left(T^{\frac{d_C+d_I+1}{d_C+d_I+2}}(\ln T)^{\frac{1}{d_C+d_I+2}}\right)$	$\mathcal{O}\left(T^{\frac{2d_C+d_I+2}{d_C+d_I+2}}(\ln T)^{\frac{d_I+2}{d_C+d_I+2}}\right)$	$\mathcal{O}\left(T^{\frac{d_C+d_I}{d_C+d_I+2}}(\ln T)^{\frac{2}{d_C+d_I+2}}\right)$
ETDC	Yes	$\mathcal{O}\left(T^{\frac{d_C+d_I+1}{d_C+d_I+2}}(\ln T)^{\frac{1}{d_C+d_I+2}}\right)$	$\mathcal{O}(T^2)$	$\mathcal{O}(T)$

TABLE 2: Detailed information of the dataset

Category	Feature	Details
Item Features	industry_id	anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc.
	discipline_id	anonymized IDs represent disciplines such as "Consulting", "HR", etc.
	is_paid	indicates that the posting is a paid for by a company
	career_level	1-6 indicates Student to Senior Executive (0 = unknown)
	country	code of the country in which the job is offered
	latitude	latitude information
	longitude	longitude information
	region	type of place of the job
	employment	1-6 indicates full-time to voluntary (0 = unknown)
	created_at	a unix time stamp timestamp representing the time when the interaction got created
	title	concepts that have been extracted from the title of job posting
	tags	concepts that have been extracted from the tags, skills or company name
User Features	job_roles	numeric IDs that were extracted from the user's current job titles
	career_level	1-6 indicates Student to Senior Executive (0 = unknown)
	discipline_id	anonymized IDs represent disciplines such as "Consulting", "HR", etc
	industry_id	anonymized IDs represent industries such as "Internet", "Automotive", "Finance", etc
	country	describes the country in which the user is currently working
	region	specific region in the country
	experience_n_entries_class	identifies the number of CV entries that the user has listed as work experiences
	experience_years_years_experience	number of years of work experience that the user has
	experience_years_in_current	number of years that the user is already working in her current job
	edu_degree	1-3 indicates bachelor to phd (0 = unknown)
	edu_fieldofstudies	separated fields of studies that the user studied
	wtcj	the user's willingness to change jobs
	premium	the user subscribed to XING's payed premium membership

- HCT [32]: This is a context-free $\mathcal{X}$-armed bandit algorithm which can simulate the recommendation on large-scale dataset.
- ACR [33]: This is a context-aware bandit algorithm which requires knowing the total number of item ahead and constructs an item tree offline.
- ITFACP [34]: The algorithm is designed with big-data support in our prior work but its partition method for the context space is static.

Similar to the works in [28], [33], we use the offline dataset to simulate and evaluate the online algorithm without introducing bias. When simulating the abovementioned algorithms, we randomly pick a user from our database at each round and get the recommended item predicted by those algorithms. Since our dataset is very large (more than 300 million instance), for the user, we are possible to find a lot of related interaction logs. Later we can judge whether the user satisfies the item based on the interaction logs and record the reward.

D. RESULTS AND ANALYSIS:

We analyze and evaluate our algorithm in following steps:

1) Comparison with Existing Algorithms

We first compare our algorithm (ETDC) with previous works [31]–[34]. We fix the maximum number of items for each algorithm as 500,000 with $d_X = 8$. For context-aware algorithms: [33], [34] and ETDC, we set $d_C = 10$ and select 10 features to construct the context space. Then, we compare the accuracy of each algorithm by the cumulative regret and the average regret over time. The results is shown in Fig. 5 and Fig. 6. From them, we can observe that the cumulative regret of context-free algorithm i.e., UCB and HCT, is nearly linear with the increase of the number of round, which indicates that they can hardly reach the optimal policy due to neglecting the diversity of users. Specifically, at round 100,000, the regret of ACR, ITFACP and ETDC on UCB and HCT are $66.8\%, 47.3\%, 37.8$ and $94.1, 66.8\%, 53.4\%$, which decrease to $52.2\%, 42.2\%, 27.5\%$ and $69.9\%, 56.6\%, 36.9\%$ at round 200,000 repectively. The comparison indicates the significance of introducing the context information. In context-aware algorithms, we can observe that the cumulative regret of ACR, ITFACP and ETDC converge at certain rounds, among which ETDC converges fastest and reaches optimal policy at around 100,000 rounds. With the number of rounds increasing, the difference of regret between ITFACP and

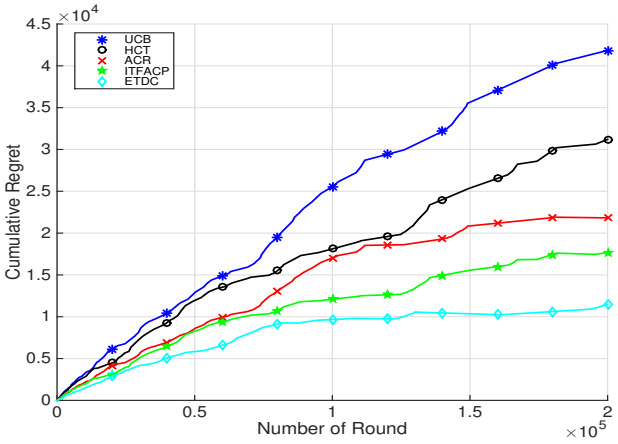


FIGURE 5: Cumulative regret of different algorithms

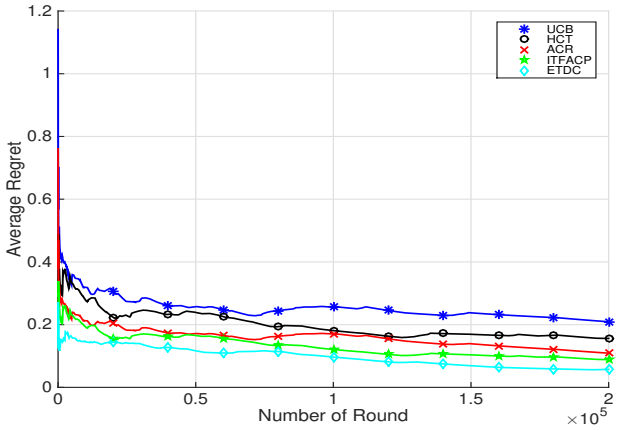


FIGURE 6: Average regret of different algorithms

ETDC increases. This is because the statistically partition method adopted by ITFACP incurs higher deviation with the increasing diversity of users. For specific data, at round 100,000, the regret of ETDC is 88.7% of what of ITFACP and it drops to 65.2% at round 200,000, which shows that ETDC performs better when facing a large-scale incoming users.

2) Scalability of Algorithms

We test the scalability of our algorithms in two aspects. We first test the influence of the size of items and compare ETDC with the benchmark algorithms. We fix the number of execution times of each algorithm at 200,000 and vary the number of items from 100,000 to 500,000 with the increment of 50,000 at each time. The result is shown in IV and Fig 5. From them, we can observe that the performance of UCB degenerates with the size of items increasing. This is because UCB individually explore items. When the size of items is more than its exploration times (rounds), the UCB cannot effectively exploit the items with high historical reward and thus its regret increase with the size of items expanding. Since ACR, ITFACP and ETDC explore items in a cluster level, their performances do not degrade. In the Fig 7, we can also find another characteristic that the curves of ETDC tends to fall when the size of item reaches 300,000. The reason is that the increasing number of items makes item space more condense and therefore the expanding of item-tree structure is more even, which enables ETDC converge to optimal policy faster. In the result, our algorithm performs better than ACR and ITFACP since our algorithm is designed for the big-data items and implements dynamic context partition. For more specific data, the regret of our algorithm is 59.6% and 66.6% of ACR and ITFACP with 400,000 items and 54.6% and 62.8% of ACR and ITFACP with 500,000 items.

Since only ETDC supports the increasing items in the process of online learning, we test its performance with different size of items adding at each round. We run ETDC 100,000 rounds with different rate of adding 100,000 new items to different size of base items. Base items are what ETDC contains before running. The rate of adding items are: 1 new item at a time with 100,000 times, 10 items at a time with 10,000 times, 100 items at a time with 1000 times. Note that the adding operations are evenly distributed in 100,000 rounds. For instances, we add 10 new items at a time at every 10 rounds and add 100 new items at a time at every 100 rounds. Therefore, more and more new items arrive with ETDC running. We set the number of base items at 1,000 and 100,000 and run ETDC 100,000 rounds. The performances are illustrated in Fig. 8 and Fig. 9, respectively. In Fig. 8, we can observe that when the number of base items (1,000) is much smaller than the number of incoming items (100,000), the influence of new arrivals of items is non-trivial, the performance degrades with the rate of incoming items increasing. This is reasonable since the high rate of incoming items (compared with items base) may greatly enlarge the diameter of nodes, which makes Assumption 1 no longer hold. Thus, the item tree cannot grow in geometric way, which hinders the convergence of ETDC. However, when the item base is large, as is shown in Fig. 9, the influence of the new arrivals is trivial. Specifically, at round 100,000, the regret of 1 items/time, 10 items/time and 100 item/time are only 101.1%, 108.3% and 114% of that of static dataset. In conclusion, ETDC has a strong scalability when the item size is static or the rate of incoming items are temperate with respect to the item base. In big-data scenario, the items base can be practically tremendous, therefore, ETDC has a strong scalability in this case.

3) Impact of d_C

Since d_C is an important parameter in context-aware algorithms, we experimental analyze its influence in this step. First, we compare the regret of ETDC with different d_C. We select features from Table. II to construct context space and vary the d_C from 2 to 13. The total number of items is fixed at 100,000 and $d_I = 5$. From Fig. 10, we can obtain that when d_C is small, the performance of ETDC is unsatisfied.

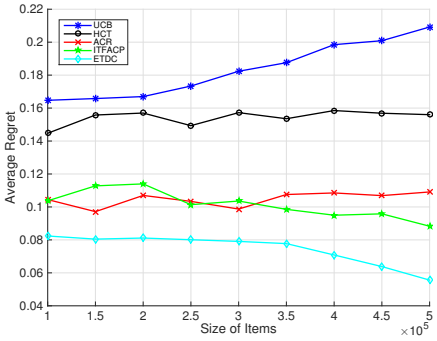


FIGURE 7: Regret of different size of items

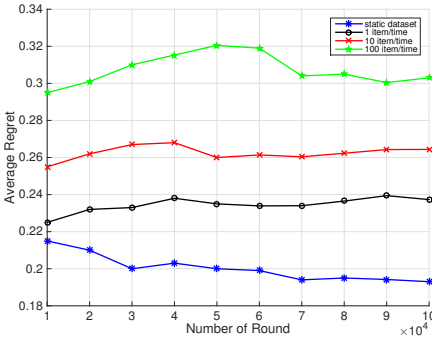


FIGURE 8: Regret of different rate of adding new items (base items are 1,000)

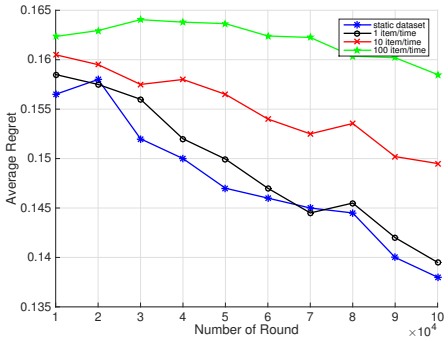


FIGURE 9: Regret of different rate of adding new items (base items are 100,000)

TABLE 3: Variation of regret with different sequence of arrivals

Variation of regret / Sequences / Rounds	Sequence1	Sequence2	Sequence3	Sequence4	Sequence5	Sequence6	Sequence7	Sequence8
10,000	**5.6%**	-4.6%	-3.6%	5.2%	-0.6%	2.8%	-4.1%	-0.6%
20,000	4.1%	-2.8%	-3.2%	**5.4%**	-2.7%	2.6%	-4.6%	1.2%
30,000	4.3%	-3.2%	-2.6%	**4.6%**	-3.6%	**4.6%**	-3.6%	1.6%
40,000	**3.6%**	-3.3%	-2.7%	3.4%	-2.7%	4.2%	-3.2%	0.7%
50,000	2.6%	-2.4%	-2.5%	**3.6%**	-3.5%	3.6%	-1.9%	0.5%
60,000	2.2%	-2.0%	-2.6%	3.2%	-3.0%	**3.8%**	-1.5%	-0.1%

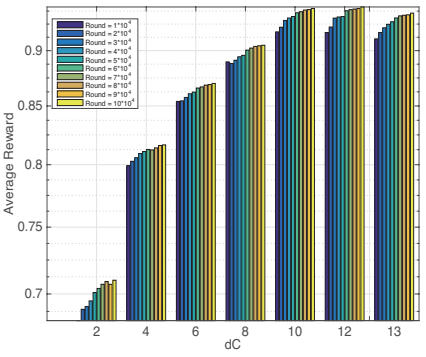


FIGURE 10: Average Reward of different d_C

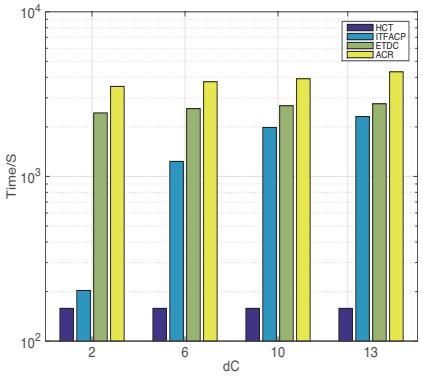


FIGURE 11: Computational cost with different d_C

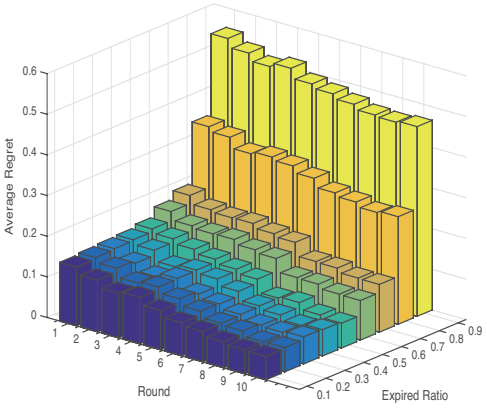


FIGURE 12: Average regret with different ratio of expired jobs

TABLE 4: The parameters on the algorithm

Parameter	Type	Value				
ρ	parameter	0.50	0.60	0.71	0.80	0.90
	regret	0.097	0.094	0.091	0.093	0.098
c	parameter	0.600	0.700	0.808	0.900	1.000
	regret	0.099	0.101	0.091	0.092	0.096
v_1	parameter	0.10	0.20	0.30	0.40	0.50
	regret	0.093	0.090	0.091	0.092	0.094

This is because the partitions of context space are not specific enough for ETDC to guarantee personalized recommendation with too few features. With the d_C increasing, the rate of the increase of average reward slows down till $d_C = 12$. Specifically, from $d_C = 2$ to $d_C = 4$, the gain of reward at 100,000 round is 15%, while from $d_C = 10$ to $d_C = 12$ the gain declines to 2.1%). Interestingly, when d_C reaches 13, the average reward of the ETDC reduces, which is a bit counterintuitive, since more features should have led to more precise clustering of users. However, as is analyzed in Section III-B, the abuse of features renders the sparsity of context space, which makes less historical rewards can be used to score the item and thus enables the historical rewards cannot be fully used. Moreover, with too many features, the context information which is trivially related to the task may also bring about the inaccurate recommendation.

Then, we select different d_C to compare the computational

cost of ACR, HCT, ITFACP and ETDC where the execution times of each algorithm is 100,000 and the size of items is 500,000. In Fig. 11, we can obtain HCT has the least computation cost due to its context-free property. Also, we find that when d_C is small ($d_C = 2$) the computation cost of ITFACP approaches that of HCT and are much lower than ACR and ETDC, which seemingly shows the outstanding scalability of ITFACP. However, as discussed above, too few features result in great deviation. Therefore, to achieve better performance, we set d_C comparatively higher to guarantee the sufficient context information to achieve personalized recommendation. With the d_C increasing, the computational cost of ITFACP sharply degenerates because the time complexity of ITFACP is $\mathcal{O}(T^{\frac{2d_C+d_I+2}{d_C+d_I+2}}(\ln T)^{\frac{d_I+2}{d_C+d_I+2}})$, which degenerates to $\mathcal{O}(n)$ with d_C increasing. From the comparison above, we conclude that the selection of d_C should be temperate, neither too many nor too few, which proves what we have discussed in theoritical part. In summary, both considering the accuracy and computational cost in practice, ETDC has the best performance.

4) Impact of expired items

In this part, we evaluate the performance of our algorithm on the dataset contained expired items. We fix the total number of items as 500,000 and adjust the ratio of expired items in the dataset. For instance, there are 400,000 unexpired items and 100,000 expired items when the expired ratio is 0.2, and the 100,000 expired items are randomly sampled and tagged from the original 500,000 items. We range the ratio between 0.1 to 0.9 and test the performance of the ETDC. From the Fig. 12, we obtain that when the ratio of expired job increase, at first the average regret increases slowly from 0.1 to 0.5. This is because the ETDC scores and analyzes the items in a cluster level, therefore as long as each cluster i.e., each leaf node contains active items, the algorithm can still efficiently converge to the best policy. However, when the expired ratio is too high (from 0.6 to 0.9) the performance of ETDC degrades dramatically. This is because the expired items plays a more important role in the dataset and most of the leaf nodes are actually empty (contain no active items).

5) Robustness of ETDC

In this step, we test the robustness of ETDC in two aspects. We first evaluate the performance of ETDC when input parameters vary. We vary ρ, c and v_1 around the optimal value and test the regret of ETDC respectively. We set the size of items as 500,000 and the execution times as 50,000 to plot the average regret. The selection of the values of parameters are detailed in Table. IV and the span of the input parameters is 0.4. From Table. IV, we can see that the regret is stable with the variation of the parameters. Specifically, the regret difference is less than 8.8% when the parameters change in the span of 0.4. This is because both the parameters ρ, c, v_1 correlatively influence the trade-off of learning speed (convergence rate) and the expanding rate (accuracy) of item tree. For instance, as we mentioned in Section III-B, when c is large, the learning of ETDC is fast but the system is less accurate. When c is small, the system's learning speed decreases while it achieves more accurate recommendation.

Then, we evaluate the robustness of ETDC when facing the arrivals of different types of users. We generate 8 sequences of user arrivals from the dataset randomly and evaluate the variations of regret over these sequences. To evaluate the variations of regret in different cases, we first calculate the average of regret up to rounds T of the 10 cases. Note the average of regret here is not average regret we used above. To avoid the confusion, we name it AR in the following discussion. AR is normalized to 1 and is used to compare the variation of each case. The results are shown in Table. III. For each sequence of users, a positive variation indicates the regret is higher than AR and a negative variation means the regret is lower than the AR. We can obtain from Table. III that the variation of regret up to different rounds is less than 5.6 percent, which proves the robustness of ETDC when dealing with different types of users.

V. CONCLUSION

In this paper, we formulate the job and candidate recommendation in PSNs as a context-aware online learning problem. To solve the problem, we propose an Expandingly Tree-based and Dynamically Context-aware Online Learning algorithm (ETDC). In ETDC, we formulate users' context space based on their explicit information and extract their reward (satisfaction) from their implicit information. Furthermore, to tackle with the issues in big-data scenario, we develop a tree-based model to analyze the items in the cluster level and thus enable the algorithm computationally efficient facing a large-scale items. The tree-based model is also adaptive for the new arrivals and expiration of the items. To guarantee the personalized recommendation for different users, we build the context space from the users explicit information and dynamic partition it in each round. Therefore, the rewards from the similar users can be used to score the scoring tree structure and thus relieve the cold start problem. Finally, we throughly evaluate the performance of our algorithm and compare it with other state-of-art algorithms. Since the exposure of users' context may result in the divulgence of their sensitive information, our future work will focus on how to diminish the possibly privacy leakage in the recommendation system in professional social network.

REFERENCES

[1] M. Aljukhadar, S. Senecal, and C. E. Daoust, "Using recommendation agents to cope with information overload," International Journal of Electronic Commerce, vol. 17, no. (2), pp. 41-70, 2012.

[2] S. Laumer, F. Gubler, C. Maier, T. Weitzel, "Job Seekers' Acceptance of Job Recommender Systems: Results of an Empirical Study" Proceedings of the 51st Hawaii International Conference on System Sciences, 2018.

[3] M. Reusens, W. Lemahieu, B. Baesens, and L. Sels, "A note on explicit versus implicit information for job recommendation," Decision Support Systems, vol. 98, pp. 26-35, 2017.

[4] K. Kenthapadi, B. Le, and G. Venkataraman, "Personalized job recommendation system at linkedin: Practical challenges and lessons learned.," In

Proceedings of the Eleventh ACM Conference on Recommender Systems, pp. 346-347, 2017.
[5] L. Wu, S. Shah, S. Choi, M. Tiwari, and C. Posse, "The Browsemaps: Collaborative Filtering at LinkedIn", In RSWeb@ RecSys, 2014.
[6] W. Shalaby, B. AlAila, M. Korayem, L. Pournajaf, K. AlJadda, S. Quinn, and W. Zadrozny, "Help me find a job: A graph-based approach for job recommendation at scale", IEEE International Conference on Big Data, pp. 1544-1553, 2017.
[7] A. Gupta, D. Garg, "Applying data mining techniques in job recommender system for considering candidate job preferences," Advances in Computing, Communications and Informatics ICACCI, 2014 International Conference on. IEEE, pp. 1458-1465, 2014.
[8] S. Yang, M. Korayem, K. AlJadda, T. Grainger, and S. Natarajan, "Combining content-based and collaborative filtering for job recommendation system: A cost-sensitive Statistical Relational Learning approach", Knowledge-Based Systems, vol. 136, pp. 37-45, 2016.
[9] D. Arya, G. Venkataraman, A. Grover, and K. Kenthapadi, "Candidate Selection for Large Scale Personalized Search and Recommender Systems," In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 1391-1393, ACM, 2017.
[10] V. Radevski, Z. Dika, and F. Trichet, "CommOn: A framework for developing knowledge-based systems dedicated to competency-based management", In Information Technology Interfaces, 2006. 28th International Conference on, pp. 419-424, IEEE, 2006.
[11] M. Zhao, F. Javed, F. Jacob, and M. McNair, "SKILL: A System for Skill Identification and Normalization", AAAI, pp. 4012-4018, 2015.
[12] S. Guo, F. Alamudun, and T. Hammond, "Resumatcher: A personalized resume-job matching system.", Expert Systems with Applications, vol. 60, pp. 169-182, 2016.
[13] Y. B. Fernandez, J. J. P. arias, A. G. Solla, M. R. Cabrer, and M. L. Nores. "Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems," IEEE Trans. Consumer Electronics, vol. 54, no. 2, pp. 727-735, 2008.
[14] M. Tkalcic, A. Odic, A. Kosir, and J. Tasic, "Affective labeling in a content-based recommender system for images," IEEE Trans. Multimedia, vol. 15, no. 2, pp. 391 - 400, 2013.
[15] Z. S. Chen, J. S. R. Jang, and C. H. Lee, "A Kernel Framework for Content-Based Artist Recommendation System in Music," IEEE Trans. Multimedia vol. 13, no. 6, pp. 1371-1380, 2011.
[16] N. D. Almalis, G. A. Tsihrintzis, N. Karagiannis, and A. D. Strati, "FoDRA: A new content-based job recommendation algorithm for job seeking and recruiting," in 2015 6th International Conf. Information, Intelligence, Systems and Applications, pp. 1-7, 2015.
[17] M. A. Brandão, and M. M. Moro, "Social professional networks: A survey and taxonomy," Computer Communications, vol. 100, pp. 20-31, 2017.
[18] J. Jarrett, and M. B. Blake, "Using Collaborative Filtering to Automate Worker-Job Recommendations for Crowdsourcing Services," in IEEE International Conf. Web Services, pp. 641-645, 2016.
[19] S. Jiang, X. M. Qian; J. L. Shen, Y. Fu, and T. Mei, "Author Topic Model-Based Collaborative Filtering for Personalized POI Recommendations," IEEE Trans. Multimedia, vol. 17, no. 6, pp. 907-918, 2015.
[20] X. Luo, M. C. Zhou, Y. N. Xia, and Q. S. Zhu, "An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems," IEEE Trans. Industrial Informatics, vol. 10, no. 2, pp. 1273-1284, 2014.
[21] G. Go, J. Yang, H. Park, and S. Han, "Using Online Media Sharing Behavior as Implicit Feedback for Collaborative Filtering," in 2010 IEEE Second International Conf. Social Computing , pp. 439-445, 2010.
[22] T. Li, A. Liu, and C. Huang, "A Similarity Scenario-Based Recommendation Model With Small Disturbances for Unknown Items in Social Networks," IEEE Access vol. 4, pp. 9251-9272, 2016.
[23] T. Lu, D. Pál, and M. Pál, "Contextual Multi-Armed Bandits," Journal of Machine Learning Research, vol. 9, pp. 485-492, 2010.
[24] A. Slivkins, "Contextual bandits with similarity information," Journal of Machine Learning Research, vol. 15, no. 1, pp. 2533-2568, 2009.
[25] C. Tekin, and M. van der Schaar, "Contextual Online Learning for Multimedia Content Aggregation," IEEE Trans. Multimedia, vol. 17, no. 4, pp. 549-561, 2015.
[26] C. Tekin, and M. van der Schaar, "Distributed online Big Data classification using context information," in 51st Annual Allerton Conf. Communication, Control, and Computing, Allerton, pp. 1435 - 1442, 2013.
[27] A. Slivkins "Contextual bandits with similarity information", Proceedings of the 24th annual Conference On Learning Theory, pp. 679-702 2011.
[28] L. Li, W. Chu, J. Langford, and R. Schapire, "A contextual-bandit approach to personalized news article recommendation," International Conference on World Wide Web ACM, pp. 661-670, 2010.
[29] L. Wang, C. Wang, K. Wang, and X. He, "BiUCB: A Contextual Bandit Algorithm for Cold-Start and Diversified Recommendation", In Big Knowledge (ICBK), 2017 IEEE International Conference on, pp. 248-253, 2017.
[30] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvari, "X-armed bandits," Journal of Machine Learning Research, pp. 1655-1695, 2011.
[31] Auer, Peter, N. C. Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem." Machine Learning, vol. 47, no. 2, pp. 235-256, 2002.
[32] M. G. Azar, A. Lazaric, and E. Brunskill, "Online Stochastic Optimization under Correlated Bandit Feedback," ICML, pp. 1557-1565, 2014.
[33] L. Q. Song, C. Tekin, and M. van der Schaar, "Online Learning in Large-scale Contextual Recommender Systems," IEEE Trans. Services Computing, vol. 19, no. 3, pp. 433-445, 2014.
[34] S. Dong, Z. Lei, P. Zhou, K. Bian and G. Liu, "Job and Candidate Recommendation with Big Data Support: A Contextual Online Learning Approach," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, pp. 1-7 2017.
[35] E. R. Núñez-Valdéz, J. M. C. Lovelle, O. S. Martínez, V. Garcia-Diaz, P. O. De Pablos, and C. E. M. Marin, "Implicit feedback techniques on recommender systems applied to electronic books", Computers in Human Behavior, vol. 28, no. 4, pp. 1186-1193, 2012.
[36] Y. Hu, Y. Koren and C. Volinsky, "Collaborative filtering for implicit feedback datasets", Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), pp. 263-272, 2008.
[37] R.Rafter and B.Smyth, " Passive profiling from server logs in an online recruitment environment", In Workshop on Intelligent Techniques for Web Personalization at the the 17th International Joint Conference on Artificial Intelligence, Seattle, Washington, USA, 2001.
[38] M. Reusens, W. Lemahieu, B. Baesens, and L. Sels, "A note on explicit versus implicit information for job recommendation", Decision Support Systems, vol. 98, pp. 26-35, 2017.
[39] S. Shalev-Shwartz, "Online learning and online convex optimization", Foundations and Trends in Machine Learning, vol. 4, no. 2, pp. 107-194, 2012.
[40] W. Chen, P. Zhou, S. Dong, S. Gong, M. Hu, and K. Wang, "Supplementary of Tree-based Contextual Learning for Online Job or Candidate Recommendation with Big Data Support in Professional Social Networks". [Online]. Available: https://www.dropbox.com/s/yd85l2ra5ndndan/AccSup.pdf?dl=0

WENBO CHEN (S'16) is currently pursuing the PhD degree in School of Electronic Information and Communications at Huazhong University of Science and Technology, Wuhan, China, working with Prof. Pan Zhou. His research interests are in the area of big data, privacy and machine learning.

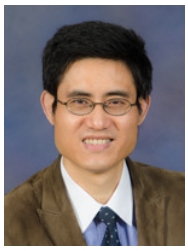
PAN ZHOU (S'07–M'14) is currently an associate professor with School of Electronic Information and Communications, Wuhan, P.R. China. He received his Ph.D. in the School of Electrical and Computer Engineering at the Georgia Institute of Technology (Georgia Tech) in 2011, Atlanta, USA. He received his B.S. degree in the Advanced Class of HUST, and a M.S. degree in the Department of Electronics and Information Engineering from HUST, Wuhan, China, in 2006 and 2008, respectively. He held honorary degree in his bachelor and merit research award of HUST in his master study. He was a senior technical member at Oralce Inc., America, during 2011 to 2013, and worked on Hadoop and distributed storage system for big data analytics at Oralce Cloud Platform. His current research interest includes: big data analytics and machine learning, security and privacy, and information networks.

SHAOKANG DONG (M'15) obtained his B.S degree in the Advanced Class of HUST, Wuhan, China, in 2014. He is currently a Ph.D student in the Department of Computer Science from Nanjing University. His research interests include machine learning, transfer reinforcement learning and deep reinforcement learning.

SHIMIN GONG (M'15) received the B.E. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, China, in 2009, and the Ph.D. degree in computer engineering from Nanyang Technological University, Singapore, in 2014. He is currently an Associate Professor with the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He was a visiting scholar at The Chinese University of Hong Kong, Shatin, Hong Kong in 2011, and the University of Waterloo, Waterloo, ON, Canada, in 2012. His research interests include wireless powered D2D networks, mobile edge computing, backscatter communications and networking.

MENGLAN HU received the BE degree in electronic and information engineering from Huazhong University of Science and Technology, China, in 2007, and the PhD degree in electrical and computer engineering from the National University of Singapore, Singapore, in 2012. He is currently an Associate Professor at the School of EIC, Huazhong University of Science and Technology, China. His research interests includes cloud computing, parallel and distributed systems, scheduling and resource management, as well as wireless networking.

KEHAO WANG received the BS degree in Electrical Engineering, MS degree in Communication and Information System from Wuhan University of Technology, Wuhan, China, in 2003 and 2006, respectively, and Ph.D in the Department of Computer Science, the University of Paris-Sud XI, Orsay, France, in 2012. From Feb. 2013 to Aug. 2013, he was a postdoc with the HongKong Polytechnic University. In 2013, he joined the School of Information Engineering at the Wuhan University of Technology, where he is currently an associate professor. From Dec. 2015, he has been a visiting scholar in the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA. His research interests include: stochastic optimization, operation research, scheduling, wireless network communications, and embedded operating system.



DAPENG WU (S'98-M'04-SM06-F'13) received Ph.D. in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, PA, in 2003. He is a professor at the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, data mining and information and network security.