This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

Recommendation for Repeat Consumption from User Implicit Feedback

Jun Chen, Chaokun Wang, Jianmin Wang, and Philip S. Yu, *Fellow, IEEE*

Abstract—Recommender system has been studied as a useful tool to discover novel items for users while fitting their personalized interest. Thus, the previously consumed items are usually out of consideration due to the "lack" of novelty. However, as time elapses, people may forget those previously consumed and preferred items which could become "novel" again. Meanwhile, repeat consumption accounts for a major portion of people's observed activities; examples include: eating regularly at a same restaurant, or repeatedly listening to the same songs. Therefore, we believe that recommending repeat consumption will have a real utility at certain times. In this paper, we formulate the problem of recommendation for repeat consumption with user implicit feedback. A time-sensitive personalized pairwise ranking (**TS-PPR**) method based on user behavioral features is proposed to address this problem. The proposed method factorizes the temporal user-item interactions via learning the mappings from the behavioral features in observable space to the preference features in latent space, and combines users' static and dynamic preferences together in recommendation. An empirical study on real-world data sets shows encouraging results.

Index Terms—Repeat Consumption, User Implicit Feedback, Time-Sensitive Personalized Pairwise Ranking.

1 Introduction

The past two decades have witnessed the fast development of recommender systems in both research fields and industrial companies [1], [2], [3]. The basic idea of building a recommender system is to design a proper utility function which can predict how much a user prefers an item [4]. From a conventional viewpoint, recommender systems should only recommend the undiscovered items to the users, because it is assumed that users can always find the previously consumed items by themselves whenever they want. This is one of the reasons why *novelty* [5] and *serendipity* ("fortunate happenstance" or "pleasant surprise") [6] are also introduced as important metrics to evaluate the performance of recommender systems.

However, it is doubtful to only recommend undiscovered items. People's consumption behaviors can be characterized by a mixture of repeat and novelty-seeking behaviors [7], [8]. It is reported that repeat consumption even accounts for a greater portion of people's observed activities compared with novelty-seeking consumption. For example, there are about 77% of listening behaviors on the previously listened songs, while the rest on the songs listened for the first time in Last.fm [9]. Due to the fact that people forget about things as time elapses [10], [11], [12], it is possible that users may prefer the previously consumed items but cannot remember how to find them at certain times. For instance, Mary wants to have steak for dinner tonight and there happens to be one of her favorite steak houses in town, which she only visited once months ago and gave a high rating on. Unfortunately, Mary cannot remember it now due to memory forgetting. It would be even more difficult for her to remember, if she has also been to many other steak houses during this period. Thus, it is appropriate to recommend this previously visited steak house for Mary today. This kind of scenarios calls for *recommendation for repeat consumption* (a.k.a. *reconsumption* [13]), which is mostly neglected in the recommender system literature.

Recommendation for **R**epeat **C**onsumption (RRC) can play a key role in people's daily lives. Since repeat consumption accounts for the major traffic of people's consumption behaviors, it is useful to propose specialized recommendation methods focusing on repeat consumption, as what have been done to the novel item recommendation problem in the last two decades.

Before addressing the RRC problem, it should be clearly understood: How to define the scope of repeat consumption, or what is the candidate item set for the RRC problem? The previous work [13] studied the problem of people's **S**hort-**T**erm **REC**onsumption (STREC) behaviors. That is to predict *whether or not* the active user will reconsume what (s)he has consumed in the last N time steps (e.g. $N \leq 100$) at a given time. A truncated time window was employed for the following reasons: (1) The recency effect is generally observed in repeat consumption, where people have a tendency to reconsume what they consumed recently [7]; (2) People's preferences change dynamically, and the recent consumption behaviors are more representative of their temporal preferences; (3) By looking back a limited number of time steps, the number of *reconsumable* candidates to consider is reduced, which is beneficial to fast online algorithms. Similarly, it is also important to study the RRC problem over a time window, i.e. to recommend items which have been consumed by the active user in the last N steps.

The difference between the RRC problem in this study

- *J. Chen, C. Wang and J. Wang are now with School of Software, Tsinghua University, Beijing 100084, China.*
- *P.S. Yu is now with Department of Computer Science, University of Illinois at Chicago, IL 60607, USA.*
- *Email: chenjun14@mails.thu.edu.cn, chaokun@tsinghua.edu.cn, jimwang@tsinghua.edu.cn, psyu@uic.edu.*

Manuscript received June 28, 2015; revised October 20, 2015.

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

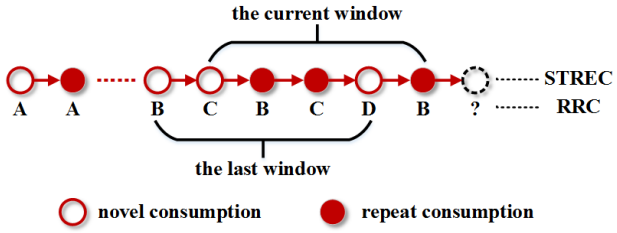


Fig. 1. An illustration of the RRC and the STREC problems. People's consumption behaviors are represented by circles along timeline. Empty circles are novel consumption behaviors, while solid circles are repeat consumption behaviors defined over a time window (e.g. capacity is 5). Characters under circles represent different items. The STREC problem is to predict whether the next incoming circle (dashed) will be solid or empty. The RRC problem is to recommend items from the current time window if the next incoming circle is predicted to be solid.

and the STREC problem in the previous work [13] is illustrated with an example in Fig. 1. In this paper, *novel consumption* is performed on the item which has not been consumed by the active user in the last N time steps. In contrast, repeat consumption is performed on the item which appears in the time window right before this consumption. For example, in Fig. 1, item B appears in the *last window*, and then the consumption of item B at the last observed position is a repeat consumption. The STREC problem is to predict whether or not the next incoming consumption is a repeat consumption given the *current window*. However, the RRC problem is to recommend items from the *current window* when the next incoming consumption is predicted to be a repetition. The study on the STREC problem [13] acts as a *switch* that divides the recommendation problem into the classical novel item recommendation problem [8], [14], [15] and the RRC problem.

In this paper, we attempt to address the RRC problem by proposing a time-sensitive personalized pairwise ranking model based on the behavioral features extracted from user implicit feedback in the consumption history. User implicit feedback is considered rather than explicit feedback, because users seldom give repeated explicit feedback (e.g. rating, comment) on a same item unlike giving implicit feedback such as clicking, viewing, listening and location check-in. To deal with this problem, we are confronted with several challenges: (1) There is lack of evidence about what are the essential factors influencing people's repeat consumption behaviors; (2) The dynamics of temporal user preferences increase the difficulty in recommendations; (3) Only limited information can be obtained from user implicit feedback, especially when domain-specific data is not available. The main contributions of this paper can be summarized as follows:

- We formulate the problem of recommendation for repeat consumption instead of novel consumption. This problem is quite unexplored and is usually neglected in the recommender system literature.
- We bring forward a **T**ime-**S**ensitive **P**ersonalized **P**airwise **R**anking (**TS-PPR**) model using domain-independent behavioral features extracted from user implicit feedback to deal with the RRC problem. This model factorizes the temporal user-item interactions via learning the mappings from the behavioral features in observable space to the preference features in latent space, and combines users' static and dynamic preferences together in recommendation.
- We introduce some generic behavioral features into the proposed method. Through experiments, we find these features important in reflecting people's preferences of repeat consumption. Meanwhile, more domain-specific features can also be appended to the vector representation of behavioral features as extensions.
- We show that **TS-PPR** is more effective than all the baselines through experimental evaluation on two real-world data sets. Besides, the sensitivity of parameters and the efficiency evaluation on the proposed method are presented.

2 Related Works

2.1 Repeat Consumption Behaviors in the Web

There are many types of repeat consumption behaviors from Web users. We show some studies on them here.

Web revisitation has been well studied in recent years [16], [17], [18], [19], [20], [21], [22]. Adar *et al*. [17] classified web revisitation behaviors into different groups based on repeat frequencies. The fast group contains web pages revisited by minutes such as hub-and-spoke pages and shopping sites. The medium group consists of hourly revisited pages like popular search engine homepages and web forums. Besides, the daily or even longer revisited pages like weekend activity pages and children-oriented pages, comprise the slow revisitation group. To facilitate web revisitation, contextual suggestions are proposed to remind users of the previously visited pages which are related to the currently viewing one [20], [21].

Repeat query is another type of web repeat consumption behavior. The repeat query may be lexically different from the original query. But as long as the user's search intent does not change, e.g. clicking the same URLs, the latter query can still be considered as the repetition and it is usually better than the initial query [23]. In addition, the short-term repeat queries are found in the web search resulting from hitting the "Next" and the "Back" buttons, refreshing the page, or automatically re-submitting the query [24], [25].

Information re-finding is also a challenging topic in people's repeat consumption behaviors [26]. There exist many effective strategies in information re-finding such as bookmarks and annotations [27]. To facilitate the effectiveness in information re-finding, Deng *et al*. [28], [29] constructed a context memory model to represent people's memory using a contextual hierarchy, and then they used a recall-by-context method to perform information re-finding.

2.2 Repeat Consumption Prediction

Recently, there is an increasing trend to study general repeat consumption behavior with user implicit feedback [7], [9], [30], [31], [8]. The dynamics of repeat consumption was studied in [7], in which the authors found that people generally tend to reconsume items with high *quality* (e.g. frequency) and high *recency* (recently consumed). They proposed a mixed weighted method for prediction, which learned the latent weights of item quality and recency gap by maximizing the value of a log-likelihood function. In [31], Kapoor *et al* studied how people *enter* the state of repeatedly

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

listening to the songs from a same artist and how they *exit* the state using the survival analysis approaches. Then, these authors used the Cox's proportional hazard model to predict when the given user will return to the service after a gap [30]. This problem is similar to RRC, except that it [30] works on continuous time scale with domain-specific features. In [9], the authors used two latent states to model repeat consumption and novel consumption, and they claimed that the transitions between the two states are driven by the level of boredom that user feels in activity stream. Thus, they proposed a hidden semi-Markov model based on these two states to predict user return time. In [8], [32], a hidden Markov model was used to recommend regular point-of-interest by estimating the prior probability of locations as well as the emission probability from locations to discretized time like days and hours. In the previous work on STREC [13], Chen *et al*. extracted four behavioral features and proposed a linear Lasso method and a quadratic method to predict whether or not a repeat consumption occurs at a given time. Basically, these works are dealing with a user revisiting time prediction problem or an analysis problem on repeat consumption behaviors rather than an item recommendation problem.

2.3 Temporal Recommendation with Implicit Feedback

User implicit feedback (e.g. click, view, listening) is more common than explicit feedback (e.g. rating, comment) in our daily lives. Thus, recommendation methods based on user implicit feedback are also widely studied [33], [34], [35], [36], [37], [38]. Basically, this recommendation problem can be solved with matrix factorization by replacing the rating matrix with a binary-valued matrix [37], and learning latent user/item features using the alternative least square algorithm. Due to the fact that implicit feedback is usually binary, many one-class collaborative filtering methods are proposed to deal with this recommendation problem [35], [39], [40]. When time information is available, the methods based on Markov Chain can be used to tackle this problem [41], [14]. In addition, learning-to-rank methods are widely used in recommendation with implicit feedback. The Bayesian personalized pairwise ranking method [33] and its extension [36] are the state-of-the-art in this category. They attempt to learn a personalized ranking function based on user implicit feedback to rank the priority of each item in the recommendation list.

The temporal effects in recommendation are widely studied. In [42], the authors proposed a coupled tensor factorization model by filling the entries in the {user-item-time} tensor with the number of user-item interactions in a given time bin. Meanwhile, it also incorporated the side information like user demographics, by sharing the latent features with the proposed tensor. Although [42] and its variant [43] considered the repeat consumption behaviors by counting the frequency of user-item interactions in the tensor entries, they are not focusing on recommending items for repeat consumption. Instead, similar to [41], their recommendation lists are the mixtures of novel items and repeat items. Another difference to the proposed method in this work lies in the processing of time. The authors in [42], [43] divide continuous time into time bins to construct the tensor. On the contrary, the consumption time points/steps like [7] are used in this paper. The similar definition of tensors considering temporal dynamics can also be found in [44], [45]. In [44], the authors combine the CANDECOMP/PARAFAC tensor decomposition method with the truncated Katz scores to predict user's temporal preference. The recommendation results do not separate repeat items from novel items just like [41], [42], [43]. In [45], a personalized transition matrix between latent user features is proposed, which is estimated by minimizing the L2 loss of predicted ratings. It uses explicit ratings unlike implicit feedback used in this work. Therefore, both methods cannot deal with the RRC problem although they model user's temporal preferences. The temporal effects in location recommendation are also studied [46], [47]. Users' repeat visit counts are used to construct the {user-item-time bin} tensor. The recommendation results are generated by performing tensor factorization with constraints like temporal consecutiveness [46], regularity and conformity [47].

However, these recommendation methods do not focus on repeat consumption, and cannot be directly used in the RRC problem. The RRC problem we attempt to tackle is quite unexplored, and it would broaden the applications of recommender system research.

3 Problem Definition

Suppose that each user $u \in \mathcal{U}$ is observed with a consumption sequence $\mathbf{S}_u = \{x_1^u, ..., x_t^u, ...\} \in \mathcal{S}$, $\forall x_t^u \in \mathcal{V}$, where $\mathcal{V}$ is the whole item set, $\mathcal{S}$ is the set of all consumption sequences from all users, and x_t^u is the consumption behavior of user u at time step t. Similar to [14], [13], [7], [41], [48], t is only used to index the position of a consumption behavior in a time-ascending consumption sequence. Please note that a *sequence* is distinguished from a *set* in that a *set* is a collection of distinct elements regardless of order, while a *sequence* is an ordered list of items where repetition may exist. In this paper, we deal with user implicit feedback only, i.e. we consider x_t^u as a positive feedback on a certain item. The elements in $\mathbf{S}_u$ are sorted in a time ascending order, i.e. consumption x_t^u happens before $x_{t+\Delta}^u$, $\forall t, \Delta \in \mathbb{N}_+$. In this paper, "time" is represented by discrete consumption step for simplicity.

To better describe the RRC problem, we first formally define a "time window" as:

Definition 1 (Time Window). Let $\mathbf{W}_{ut}$ be a subsequence of $\mathbf{S}_u$ ending on the consumption behavior x_t^u, i.e. $\mathbf{W}_{ut} = \{x_{t-|\mathbf{W}_{ut}|+1}^u, ..., x_t^u\} \subseteq \mathbf{S}_u$. $\mathbf{W}_{ut}$ is called a time window w.r.t. user u and time t.

Time window $\mathbf{W}_{ut}$ contains the last $|\mathbf{W}_{ut}|$ consumption behaviors from u at the time step t. In this paper, we use $\mathbf{W}$ to denote a general time window. Then, we can define the RRC problem as:

Definition 2 (RRC Problem). Given the time window $\mathbf{W}_{ut}$ w.r.t. user u and time t, if the next incoming unknown consumption behavior x_{t+1}^u is predicted very likely to be a repetition from $\mathbf{W}_{ut}$ (i.e. $x_{t+1}^u \in \mathbf{W}_{ut}$), then the RRC problem is to recommend items $v \in \mathbf{W}_{ut}$ to u, which satisfy u's personalized preference. [1]

1. If not a repetition, then it becomes the classical novel item recommendation problem.

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

TABLE 1
Description of symbols.

Symbol	Description
$\mathcal{U}, \mathcal{V}, \mathcal{S}$	user / item / sequence sets
u, v	an arbitrary user / item
$\mathbf{S}_u$	consumption sequence of user u
x	a random variable of item
$\mathbf{W}$	a time window
$\mathbf{u}, \mathbf{v}$	user / item latent feature vectors
$\|\cdot\|$	cardinality of a set or sequence
$\|\cdot\|_F$	Frobenius-norm
$\mathbf{A}_u$	feature transform matrix of user u
K	dimension of latent feature space
F	dimension of observable feature space
S	# negative samples for each positive sample
γ, λ	regularization parameters
Ω	the minimum gap in the evaluation

It is clear that one of the major differences between the RRC problem and the classical novel item recommendation problem is the candidate set. The candidate items for RRC are those appearing in $\mathbf{W}_{ut}$, while the candidate set for novel item recommendation is $\mathcal{V} - \{v \in \mathcal{V} | v \in \mathbf{S}_u\}$. The RRC problem is defined over a time window because the dynamics of repeat consumption can be reflected in user's recent behaviors. In real applications, we can set an ideal time window length $|\mathbf{W}|$ based on the general gap between adjacent consumption behaviors.

The method to predict whether or not the incoming unknown consumption is a repetition is out of the scope of this paper. It may be predicted using the existing algorithms like STREC [13], or using some other auxiliary information; examples of auxiliary information include: a regular customer repurchases daily supplies on every Saturday night; a steak-lover goes to steak houses for dinner every other day. However, it is unknown what the regular customer would repurchase and which steak house the steak-lover would revisit. This is why addressing RRC has a real utility.

Although it may actually be better to somehow mix the results from RRC and novel item recommendation before presenting to users, we would like to focus on RRC in this paper, and leave the mixture problem in our future work. Thus, we assume that the next incoming consumption has already been identified to be a repetition or not. The description of the symbols used in this paper is shown in Table 1.

4 Time-Sensitive Personalized Pairwise Ranking Model for RRC

In this section, we first introduce the basic pairwise ranking model, after which the proposed optimized method to deal with the RRC problem is discussed.

4.1 Personalized Pairwise Ranking (PPR) Model

To tackle the recommendation problem with user implicit feedback, the Bayesian personalized pairwise ranking (**PPR**) model [33], [36] is widely known as the state-of-the-art. Basically, the main idea of this model is to learn a personalized pairwise ranking function $p(>_u | \Theta)$ which generates a partial order among all items. Then, the global ranking list is obtained for recommendation. Here Θ is the set of parameters in the ranking function. For example, $\forall v_i, v_j \in \mathcal{V}$, if $p(v_i >_u v_j | \Theta) > p(v_j >_u v_i | \Theta)$, then this model predicts that user u prefers v_i to v_j.

For a given triple (u, v_i, v_j), this basic model can tell which one u prefers more. Let $\mathbf{u}, \mathbf{v}_i$ and $\mathbf{v}_j$ denote the K-dimensional latent features w.r.t. the triple (u, v_i, v_j). The preference is usually computed as a scalar value for comparisons. When using inner product as preference function:

$$r_{uv} = \mathbf{u}^\top \mathbf{v}, \tag{1}$$

the pairwise ranking function can be denoted by:

$$\begin{aligned} p(v_i >_u v_j | \Theta) &= \sigma(r_{uv_i} - r_{uv_j}) \\ &= \sigma(\mathbf{u}^\top \mathbf{v}_i - \mathbf{u}^\top \mathbf{v}_j) \\ &= \sigma(\mathbf{u}^\top (\mathbf{v}_i - \mathbf{v}_j)). \end{aligned} \tag{2}$$

Here, function σ is monotonically increasing and differentiable. A good choice is the Sigmoid function: $\sigma(m) = \frac{1}{1+e^{-m}}$. Thus, the pairwise ranking function can be rewritten as:

$$p(v_i >_u v_j | \Theta) = \frac{1}{1 + e^{-\mathbf{u}^\top (\mathbf{v}_i - \mathbf{v}_j)}}. \tag{3}$$

In this basic model, Θ comprises the latent features of both users and items. In order to estimate Θ, we minimize the following negative log-likelihood function with regularization term:

$$\underset{\Theta}{\operatorname{argmin}} \sum_{(u, v_i, v_j) \in \mathcal{D}} - \ln p(v_i >_u v_j | \Theta) + \frac{\lambda}{2} \|\Theta\|_F^2, \tag{4}$$

where the training set $\mathcal{D}$ consists of the triples of known preference, i.e. $\forall (u, v_i, v_j) \in \mathcal{D}$, u is known to prefer v_i to v_j. Unfortunately, the **PPR** method cannot directly deal with the RRC problem because **PPR** is only able to learn a fixed preference order between v_i and v_j for u. However, u's preference towards v_i and v_j may change with time. u prefers v_i to v_j at some time, but prefers v_j to v_i at another time. Since temporal preference is not considered in **PPR**, **PPR** is not available in the RRC problem.

4.2 Time-Sensitive Personalized Pairwise Ranking (TS-PPR) Model

Recall that the RRC problem is defined in a dynamic scheme and items can be reconsumed for several times by a same user in Section 3. Thus, the ranking function $p(>_u | \Theta)$ should be time-sensitive as well. There is a previous extension of **PPR** to factorize the tensor of transition probability between baskets of items on time snapshots [41]. However, it only focuses on the basket-to-basket transition probability rather than exploring the time-sensitive contexts of users as what we do in this work. Meanwhile, since the latent features of user and item are all learnt with Gaussian random initializations, this also increases the uncertainty and variation of the previous model. In this paper, we decrease this uncertainty by incorporating observable time-sensitive user behavioral features.

4.2.1 Model Specification

To address the aforementioned issues, we improve the ranking model and incorporate time-sensitive user behavioral features into the optimized model. Let $\mathbf{f}_{uvt}$ denote the behavioral feature vector representing the interaction between

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

user u and item v by time point t. $\mathbf{f}_{uvt}$ can be replaced by any time-sensitive user behavioral features, which will be discussed later. We redefine the preference function as:

$$r_{uvt} = \mathbf{u}^\top \mathbf{v} + \mathbf{u}^\top \mathbf{A}_u \mathbf{f}_{uvt} = \mathbf{u}^\top (\mathbf{v} + \mathbf{A}_u \mathbf{f}_{uvt}). \quad (5)$$

It is easy to see that we add a time-sensitive personalized term $\mathbf{u}^\top \mathbf{A}_u \mathbf{f}_{uvt}$ to the original preference function in Eq. (1). In this way, we incorporate the time-sensitive features $\mathbf{f}_{uvt}$ while also preserving the static user preference $\mathbf{u}^\top \mathbf{v}$. In our method, $\mathbf{u}$ and $\mathbf{v}$ are K-dimensional latent features, while $\mathbf{f}$ is an F-dimensional observable time-sensitive behavioral feature. Thus, $\mathbf{A}_u$ is a personalized mapping from F-dimensional observable behavioral space to K-dimensional latent preference space. Please note that we do not impose constraints $K = F$ on the proposed method. The inequality of feature dimensions results from that the value of F is fixed for a specific implementation of $\mathbf{f}$, while the value of K can vary: (1) If $K < F$, $\mathbf{A}_u$ is a dimension reduction linear mapping; (2) If $K = F$, $\mathbf{A}_u$ is an equal-dimension mapping; It can be set to the identity matrix $\mathbf{I}$ for simplicity in this case; (3) If $K > F$, $\mathbf{A}_u$ maps the observable feature into higher dimensional latent feature. Since the static user/item features are also preserved in our model, the increase of K is expected to improve the performance of our model.

Therefore, we focus on learning a time-sensitive personalized pairwise ranking function defined as:

$$\begin{aligned} p(v_i >_{ut} v_j) &= \sigma(r_{uv_it} - r_{uv_jt}) \\ &= \sigma(\mathbf{u}^\top(\mathbf{v}_i + \mathbf{A}_u \mathbf{f}_{uv_it} - \mathbf{v}_j - \mathbf{A}_u \mathbf{f}_{uv_jt})) \\ &= \sigma(\mathbf{u}^\top(\mathbf{v}_i - \mathbf{v}_j + \mathbf{A}_u(\mathbf{f}_{uv_it} - \mathbf{f}_{uv_jt}))) \\ &= \frac{1}{1 + e^{-\mathbf{u}^\top(\mathbf{v}_i - \mathbf{v}_j + \mathbf{A}_u(\mathbf{f}_{uv_it} - \mathbf{f}_{uv_jt}))}}. \end{aligned} \quad (6)$$

For better understanding, we illustrate the dependencies within our model with a graphical representation in Fig. 2. Shaded circles are observable variables, while empty circles are latent variables. An F-dimensional observable feature $\mathbf{f}$ is mapped to a K-dimensional latent feature $\mathbf{y} = \mathbf{A}\mathbf{f}$. The binary observable variable r indicates whether user u reconsumes item v w.r.t. time-sensitive feature $\mathbf{f}$ or not.

4.2.2 Parameter Inference

Similar to the original optimization problem defined in Eq. (4), we reformulate our objective function with regularization terms as:

$$\begin{aligned} \mathcal{J} = \sum_{(u,v_i,v_j,t)\in\mathcal{D}} & -\ln p(v_i >_{ut} v_j) + \frac{\lambda}{2}\sum_u \|\mathbf{A}_u\|_F^2 \\ & + \frac{\gamma}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2), \end{aligned} \quad (7)$$

where $\mathcal{D}$ contains quadruples (u, v_i, v_j, t) representing that u reconsumes v_i rather than v_j at time step t. The training set $\mathcal{D}$ can be formally defined as:

$$\mathcal{D}_{u,t} = \{(u, v_i, v_j, t) | v_i = x_t^u \wedge v_i, v_j \in \mathbf{W}_{u,t-1} \wedge v_i \neq v_j\}, \quad (8)$$

$$\mathcal{D} = \bigcup_{u\in\mathcal{U}} \bigcup_{t\in\mathbb{N}_+} \mathcal{D}_{u,t}. \quad (9)$$

Please note that v_i and v_j are selected from the time window $\mathbf{W}_{u,t-1}$, which means that both v_i and v_j are the *reconsumable* candidates for u at time t.

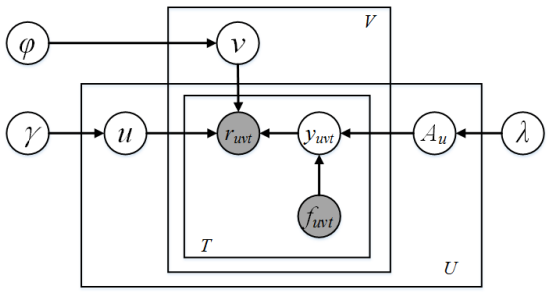


Fig. 2. A graphical representation of dependencies in **TS-PPR**.

In this model, $\mathbf{U}$ and $\mathbf{V}$ are the matrices of user and item latent preference features, respectively, while $\mathbf{A}_u$ is the personalized feature mapping of user u. λ and γ are the regularization parameters. Thus, the new optimization problem becomes:

$$\mathbf{A}^*, \mathbf{U}^*, \mathbf{V}^* = \underset{\mathbf{A}\in\mathbb{R}^{|\mathcal{U}|\times K\times F}, \mathbf{U}\in\mathbb{R}^{|\mathcal{U}|\times K}, \mathbf{V}\in\mathbb{R}^{|\mathcal{V}|\times K}}{\operatorname{argmin}} \mathcal{J}. \quad (10)$$

Basically, this optimization problem can be solved by using the stochastic gradient descent method. The gradient w.r.t. a training quadruple (u, v_i, v_j, t) is:

$$\begin{aligned} & \frac{\partial}{\partial\theta}(\ln p(v_i >_{ut} v_j) - \frac{\eta_\theta}{2}\theta^2) \\ & \quad = (1 - p(v_i >_{ut} v_j))\frac{\partial}{\partial\theta}(r_{uv_it} - r_{uv_jt}) - \eta_\theta\theta, \end{aligned} \quad (11)$$

where $\eta_\theta \in \{\gamma, \lambda\}$, $\theta \in \{\mathbf{A}, \mathbf{U}, \mathbf{V}\}$.

Specifically, we have:

$$\frac{\partial}{\partial\mathbf{u}}(r_{uv_it} - r_{uv_jt}) = \mathbf{v}_i - \mathbf{v}_j + \mathbf{A}_u(\mathbf{f}_{uv_it} - \mathbf{f}_{uv_jt}), \quad (12)$$

$$\frac{\partial}{\partial\mathbf{v}_i}(r_{uv_it} - r_{uv_jt}) = \mathbf{u}, \quad (13)$$

$$\frac{\partial}{\partial\mathbf{v}_j}(r_{uv_it} - r_{uv_jt}) = -\mathbf{u}, \quad (14)$$

$$\frac{\partial}{\partial\mathbf{A}_u}(r_{uv_it} - r_{uv_jt}) = \mathbf{u} \otimes (\mathbf{f}_{uv_it} - \mathbf{f}_{uv_jt}). \quad (15)$$

Please note that $\mathbf{u} \otimes (\mathbf{f}_{uv_it} - \mathbf{f}_{uv_jt})$ in Eq. (15) is the outer product between column vector $\mathbf{u}$ and column vector $(\mathbf{f}_{uv_it} - \mathbf{f}_{uv_jt})$.

The parameter inference algorithm using the stochastic gradient descent is shown in Algorithm 1. The latent preference features and the personalized mappings are first initialized from zero-mean Gaussian distributions with different variations. Then, it iteratively and randomly fetches a quadruple (u, v_i, v_j, t) from the training set (Lines 3—5), and updates the corresponding latent preference features and personalized mappings (Lines 6—9). To alleviate the imbalance of the numbers of repeat consumption from different users in the training, this algorithm first uniformly samples a user, then uniformly samples a repeat consumption, and finally uniformly samples a non-reconsumption accordingly. The iteration continues until the objective function converges.

However, since there could be many "negative samples" v_j for a single repeat consumption v_i ("positive sample"), it is very time-consuming to compute the time-sensitive

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

Algorithm 1 Parameter Inference Algorithm

Input:
learning rate α, regularization parameters γ, λ.
Output:
transform matrix $\mathbf{A}_u$ for each user u,
latent feature matrices $\mathbf{U}, \mathbf{V}$.

1: initialize $\mathbf{A}_u \sim N(\mathbf{0}, \lambda\mathbf{I}), \forall u$, $\mathbf{U}, \mathbf{V} \sim N(\mathbf{0}, \gamma\mathbf{I})$
2: **repeat**
3: uniformly draw a user u from user set $\mathcal{U}$
4: uniformly draw a repeat consumption of u w.r.t. item v_i at time t
5: uniformly draw item v_j ($v_j \neq v_i$) from the time window of u at time t
6: $\mathbf{u}' \leftarrow (1-\alpha\gamma)\mathbf{u} + \alpha(1-p(v_i >_{ut} v_j))\frac{\partial}{\partial \mathbf{u}}(r_{uv_it} - r_{uv_jt})$
7: $\mathbf{v}_i' \leftarrow (1-\alpha\gamma)\mathbf{v}_i + \alpha(1-p(v_i >_{ut} v_j))\frac{\partial}{\partial \mathbf{v}_i}(r_{uv_it} - r_{uv_jt})$
8: $\mathbf{v}_j' \leftarrow (1-\alpha\gamma)\mathbf{v}_j + \alpha(1-p(v_i >_{ut} v_j))\frac{\partial}{\partial \mathbf{v}_j}(r_{uv_it} - r_{uv_jt})$
9: $\mathbf{A}_u' \leftarrow (1-\alpha\lambda)\mathbf{A}_u + \alpha(1-p(v_i >_{ut} v_j))\frac{\partial}{\partial \mathbf{A}_u}(r_{uv_it} - r_{uv_jt})$
10: $\mathbf{u}, \mathbf{v}_i, \mathbf{v}_j, \mathbf{A}_u \leftarrow \mathbf{u}', \mathbf{v}_i', \mathbf{v}_j', \mathbf{A}_u'$
11: **until** $\mathcal{L}$ convergence
12: **return** $\mathbf{A}, \mathbf{U}, \mathbf{V}$

features of all the "negative samples" w.r.t. a given "positive sample" in real-time or even in offline mode. Therefore, we randomly select a limited number of "negative samples" v_j for each repeat consumption v_i, and compute their time-sensitive features in advance of training. The number of negative samples w.r.t. a single positive sample is denoted by S. Although this pre-sample strategy leads to loss of information for training, we show through experiments that the proposed algorithm with such pre-sample strategy still outperforms the baselines.

For better understanding, we use an example to illustrate the process of training quadruple pre-sample and behavioral feature extraction. Fig. 3 shows the snapshots of u's consumption history at time t, $t+1$ and $t+2$, respectively. At time t, the next incoming consumption on item v_C is a repetition since $v_C \in \mathbf{W}_t$ but v_C is not consumed within the last Ω consumption steps ($0 < \Omega < |\mathbf{W}_t|$). Parameter Ω is used to avoid trivial and less useful recommendations on the recently consumed items. In this paper, we do not consider repeat consumption from the last Ω time steps. Thus, v_C is a positive sample w.r.t. user u and time t. Then, we randomly choose negative samples in $\mathbf{W}_t$ but not within the last Ω consumption steps. Suppose we select v_A, v_B and v_D as negative samples, respectively. Then, we add quadruples (u, v_C, v_A, t), (u, v_C, v_B, t), (u, v_C, v_D, t) to $\mathcal{D}$. Meanwhile, the time-sensitive behavioral features $\mathbf{f}_{uv_Ct}$, $\mathbf{f}_{uv_At}$, $\mathbf{f}_{uv_Bt}$, $\mathbf{f}_{uv_Dt}$ are extracted. The extraction of behavioral features will be discussed shortly. However, at time $t+1$, since the next incoming consumption on item v_H is not in $\mathbf{W}_{t+1}$, it is therefore a novel consumption and we do not sample training quadruples at this time. At time $t+2$, the next incoming consumption on item v_D is a repetition again ($v_D \in \mathbf{W}_{t+2}$ but not in the last Ω steps). We randomly select some negative samples w.r.t. the positive sample v_D, and these negative samples may be v_E and v_F (Please note that some items may not be sampled such as v_C at time $t+2$.).

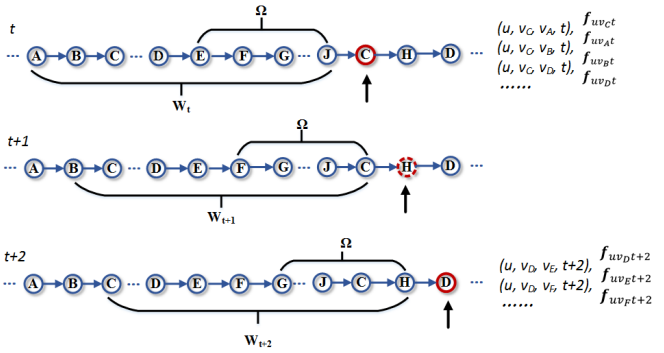


Fig. 3. An illustration of training quadruple sampling.

Then, we also add these quadruples into $\mathcal{D}$, and extract the corresponding time-sensitive behavioral features. After the training quadruples are sampled and the features are extracted, we can then perform the parameter inference.

In addition, since the size of training set $|\mathcal{D}|$ is usually very large due to the incorporation of time, the computation of the negative log likelihood $\mathcal{J}$ after each epoch in Algorithm 1 is infeasible. Instead, we can approximate $\mathcal{J}$ with the small batch approach on a fixed fraction of training quadruples, i.e. compute the objective function value $\mathcal{J}_{small}$ on n fixed quadruples after every m epochs, and check the convergence based on the value change of $\mathcal{J}_{small}$. In our experiments, we set $n = m = \frac{|\mathcal{D}|}{10}$. That is, we compute the objective function using each user's first 10% training quadruples after every $\frac{|\mathcal{D}|}{10}$ epochs. Of course, one can set other values of n, m based on the desired convergence precision and training time. Please note that this approximation does not affect the selecting probability of each training quadruples in $\mathcal{D}$. It also needs to be clarified that the small batch approach is only used for convergence check, but not for the parameter update. The corresponding parameters are updated after each epoch in the parameter inference procedure (Algorithm 1, Line 6–9).

4.3 Recommendation

Once the parameters have been inferred, we can address the RRC problem by recommending items in the current time window, which maximize the time-sensitive preference function in Eq. (5). Since the optimization goal in **TS-PPR** is to maximize the difference of preference $r_{uv_it} - r_{uv_jt}$ between the positive samples v_i and the corresponding negative samples v_j, the items with larger preference r_{uvt} should be more preferred by u at time t.

Besides the RRC problem, **TS-PPR** can be used in novel item recommendation as well. Although the number of negative samples w.r.t. each positive sample in novel item recommendation is much larger compared with that in RRC, the training quadruple pre-sample strategy can alleviate this issue. Another application of the **TS-PPR** model is to mix its results with others from a novel item recommendation method before presenting to users. This mixture can balance users' demands for both novelty-seeking and repeat consumption. However, we only focus on evaluating the performance of **TS-PPR** on the RRC problem in this paper.

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

4.4 Behavioral Feature Extraction

After introducing **TS-PPR**, we now discuss the selection of time-sensitive feature **f**. Since we are dealing with the general RRC problem, domain-specific features are out of consideration in this paper. Therefore, we only extract behavioral features from user implicit feedback.

4.4.1 Static Features

Static features represent items' intrinsic characteristics.

Item Quality. It has been shown that the item quality seriously affects people's repeat consumption behaviors [7]. Generally, the higher quality the item is, the more likely it would be reconsumed. Frequency is a good representation of item quality [7], [13]. For a given item v, we compute its normalized item quality $\bar{q_v}$ as:

$$q_v = \ln(1 + n_v), \tag{16}$$

$$\bar{q_v} = \frac{q_v - q_{min}}{q_{max} - q_{min}}, \tag{17}$$

where n_v is the frequency of item v in the training set. q_{min} and q_{max} are the minimum and the maximum q_v ($\forall v \in \mathcal{V}$) observed in the training set, respectively.

Item Reconsumption Ratio. In the previous work [13], it was found that item reconsumption ratio is a good indicator of whether or not the given user will perform a repeat consumption at a specific time. Thus, we also employ this feature in the RRC problem.

The item reconsumption ratio for v is defined as:

$$r_v = \sum_u \sum_t \frac{\mathbb{I}_{x^u_{t+1} \in \mathbf{W}_{ut} \wedge x^u_{t+1} = v}}{\mathbb{I}_{x^u_{t+1} = v}}, \tag{18}$$

where $\mathbf{W}_{ut}$ is the time window for u at time t, and $\mathbb{I}$ is an indicator function. The numerator and the denominator are the number of observations on item v as a repeat consumption and the number of total observations on it, respectively. It measures the probability of item v to be reconsumed based on the observations in the training set. Please note that the length of time window $\mathbf{W}_{ut}$ is fixed regardless of time t.

4.4.2 Dynamic Features

Dynamic features represent the time-sensitive interaction between user u and item v.

Recency Feature. Recency effect has been widely observed in people's consumption behaviors [7], [13], [14]. In [14], it is found that the hyperbolic function performs better compared with the other alternatives to measure the decay of people's interest. Thus, the hyperbolic definition of recency feature is:

$$c_{vt} = \frac{1}{t - l_{ut}(v)}, \tag{19}$$

where $l_{ut}(v)$ is u's last consumption time on item v before time t. Apparently, c_{vt} is a time-decaying feature. Another exponential alternative of recency feature is:

$$c_{vt} = e^{-(t - l_{ut}(v))}. \tag{20}$$

Dynamic Familiarity. Inspired by the previous work [13], a user may choose to reconsume the items which (s)he is familiar with. The dynamic familiarity of user u on item v in the proposed method is defined as:

$$m_{vt} = \frac{|\{x | x \in \mathbf{W}_{ut} \wedge x = v\}|}{|\mathbf{W}_{ut}|}. \tag{21}$$

The numerator is the number of consumption on item v in $\mathbf{W}_{ut}$, while the denominator is the length of the time window. This feature measures the fraction of consumption on item v in time window $\mathbf{W}_{ut}$. A larger value of m_{vt} indicates u's higher dynamic familiarity with v.

Thus, in this paper, the feature vector **f** is interpreted as $\mathbf{f} = \{\bar{q_v}, r_v, c_{vt}, m_{vt}\}^\top$. Obviously, these features have already been normalized into the range $[0, 1]$. For general purposes, we use these domain-independent features to learn the proposed recommendation model. However, the implementations of these features can be replaced by other more sophisticated and domain-specific methods.

TABLE 2
Statistics of real-world data sets.

Data Set	Type	Users	Items	Consumption
Gowalla	LBSN	14,742	936,883	4,031,705
Lastfm	Music	964	958,847	16,318,704

5 Experiments

5.1 Data Sets and Experimental Settings

In the experiments, we use two publicly available real-world data sets as follows:

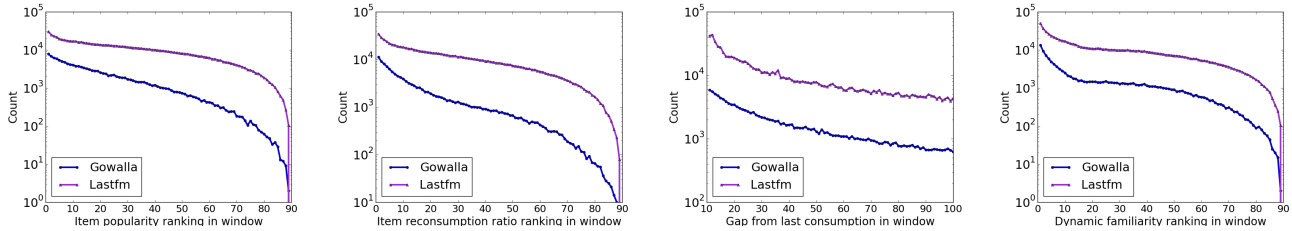
- **Gowalla** [49]. This data set contains people's check-in data on different locations in the location-based social network Gowalla. The check-in c on location l is considered as repeat consumption behavior if there are also other check-ins on location l in the time window right before c [8], [32].
- **Lastfm** [50]. This data set consists of people's listening behaviors in the website Last.fm. The listening p on song s is considered as repeat consumption behavior if there are also other listenings on song s in the time window right before p. The listening behaviors whose duration is less than 30 seconds are removed before experiments since these behaviors are considered as dislikes.

We use each user's 70% consumption sequence for training while the rest 30% for testing. Since the training requires that each user has $|\mathbf{W}| = 100$ behaviors to perform parameter inference, we only preserve the users whose consumption sequence length $|\mathbf{S}_u|$ satisfies: $|\mathbf{S}_u| \times 70\% \geq 100$. The statistics of the two data sets after filtering are shown in Table 2.

As for the evaluation, since we are dealing with a recommendation problem, we should understand what need to be recommended. Obviously, it is less needed to recommend the recently consumed items to the active user u because it is very likely that u already knows how to get those items. In another word, we should focus on recommending items which were consumed before but not in the last Ω consumption steps. In our experiments, we set $\Omega = 10$ by default, which is also evaluated in later experiments. Thus, we evaluate the performance of the proposed method on predicting the consumption $x^u_t, \forall t \in \mathbb{N}$, which is a repetition

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering



(a) Item popularity (b) Item reconsumption ratio (c) Recency (d) Dynamic familiarity

Fig. 4. The distributions of repeat consumption by the ranking of the reconsumed item in time window $\mathbf{W}$ on the four features ($|\mathbf{W}| = 100, \Omega = 10$).

from u's last $|\mathbf{W}|$ consumption steps but not in the last Ω steps ($0 < \Omega < |\mathbf{W}|$) w.r.t. time t.

To examine the significance of the extracted behavioral features in Subsection 4.4, we illustrate the distributions of reconsumed items by their rankings in the corresponding time windows on each of the four features in Fig. 4, respectively. We can see that the distributions in Fig. 4(a), That is, people tend to reconsume the items of higher item popularity, higher item reconsumption ratio and also higher dynamic familiarity. Please note that the x-axis represents the ranking of the reconsumed item on a certain feature in the corresponding time window, while the y-axis represents the count of observed repeat consumption and is plotted in the log scale. This trend is even more obvious in the Gowalla set compared with that in the Lastfm set, because the former is observed with steeper curves. Although the distribution in Fig. 4(c) is a little different from the others, it also indicates that more people would like to reconsume what they have consumed recently. Therefore, Fig. 4 has shown that the behavioral features we extracted are very representative of people's repeat consumption behaviors.

5.2 Baselines

We compare the performance of the proposed method with some baselines. These methods are:

- **Random**. This baseline randomly recommends items from the given time window. No weighting scheme on the items is used in the recommendation.
- **Pop**. This baseline generates recommendations by ranking items based on their item popularity (i.e. frequency). Item popularity was found to be an important factor in the dynamics of repeat consumption [7]. The item popularity is measured by $\ln(1 + n_v)$ where n_v is the frequency of item v.
- **Recency**. This baseline assumes that the recently consumed items are more likely to be reconsumed, which was also observed in [7]. We use an exponential recency measurement: $e^{-\Delta t_{uv}}$, where Δt_{uv} is the gap between the recommendation time point and the time point of u's last consumption on item v. This baseline recommends items weighted by $e^{-\Delta t_{uv}}$.
- **FPMC** [41]. This method employs the Tucker Decomposition on a {user-item-item} transition tensor. In the RRC problem, we adapt this method to estimate the probability of transition from a set of items (in time window) to the incoming item. It represents the group of methods based on tensor decomposition to study user's temporal preferences [42], [43], [44], [45].
- **Survival** [30]. This method based on survival analysis uses the Cox's proportional hazard function to estimate the return time for each item. Although it was previously proposed on the continuous time domain, we use it in the discrete time scale in the comparison because it has similar problem definition with the RRC. We use the Cox's proportional hazard function implemented in the Lifelines package [2].
- **DYRC** [7]. This method proposes a mixed weighted scheme to recommend repeat items based on item popularity and recency effect. It learns the latent weights of item popularity and recency gap by maximizing a log-likelihood function.

In addition, the proposed method is denoted by **TS-PPR**.

We have conducted extensive experiments to evaluate the proposed method. In the following subsections, we first present the results on accuracy performance (Subsection 5.3), followed by the experiments on the study of feature importance in **TS-PPR** (Subsection 5.4). Then, the sensitivity of parameters (Subsection 5.5) as well as the efficiency performance (Subsection 5.6) of **TS-PPR** are also presented and discussed in detail. At last, we conducted experiments to combine **TS-PPR** with the previous work **STREC** [13] as a holistic algorithm for repeat consumption behavior prediction (Subsection 5.7).

5.3 Recommendation Accuracy

First of all, we evaluate the performance on the recommendation accuracy, which is also the most important measurement. The average precision is employed as the accuracy measurement in our experiments. First, we define the recommendation precision on a single user u as:

$$P(u) = \frac{\sum_t \mathbb{I}_{x^u_{t+1} \in \mathbf{W}_{ut} \wedge x^u_{t+1} \in \mathbf{R}_{ut}}}{\sum_t \mathbb{I}_{x^u_{t+1} \in \mathbf{W}_{ut}}}, \tag{22}$$

where $\mathbf{R}_{ut}$ is the recommendation list for u at time t. Please note that the evaluation is performed on the test sequence of each user. Then, the macro average precision $MaAP$ and the micro average precision $MiAP$ are defined as:

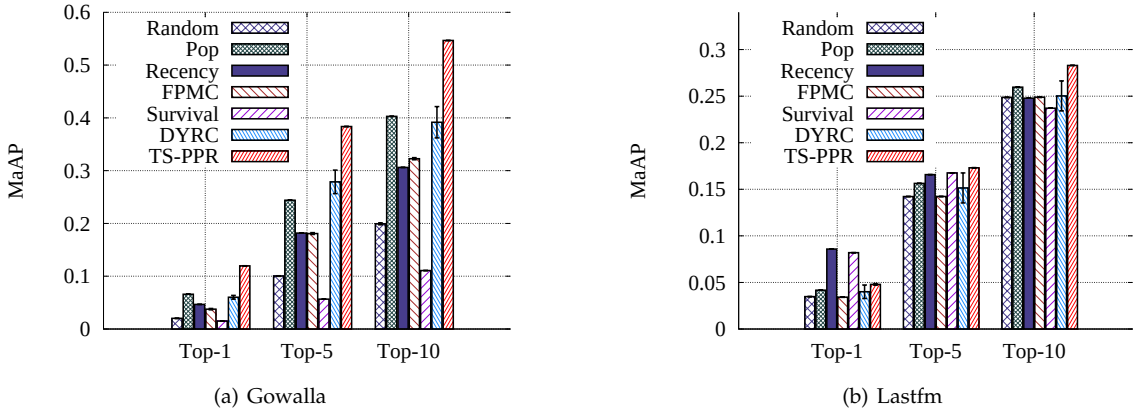
$$MaAP = \frac{\sum_{u \in \mathcal{U}} \sum_t \mathbb{I}_{x^u_{t+1} \in \mathbf{W}_{ut} \wedge x^u_{t+1} \in \mathbf{R}_{ut}}}{\sum_{u \in \mathcal{U}} \sum_t \mathbb{I}_{x^u_{t+1} \in \mathbf{W}_{ut}}}, \tag{23}$$

$$MiAP = \frac{\sum_{u \in \mathcal{U}} P(u)}{|\mathcal{U}|}. \tag{24}$$

2. https://github.com/CamDavidsonPilon/lifelines

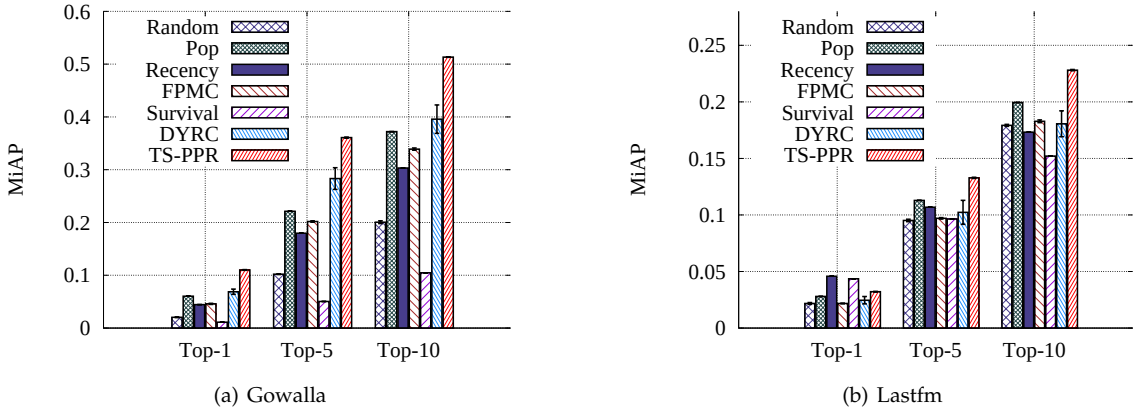
1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering



(a) Gowalla (b) Lastfm

Fig. 5. Macro average precision performance of all the methods ($\Omega = 10$, $S = 10$).



(a) Gowalla (b) Lastfm

Fig. 6. Micro average precision performance of all the methods ($\Omega = 10$, $S = 10$).

In Eq. (23), the denominator is the total number of times that a recommendation list is generated in the evaluation, while the numerator is the total number of times that a recommendation list is correct (i.e. contains the reconsumed item). Thus, $MaAP$ measures the fraction of the total correct recommendations in the evaluation. However, $MiAP$ is the mean recommendation precision over all users in the evaluation. The major difference between $MaAP$ and $MiAP$ is whether or not the imbalance of user sequence length should be considered. $MaAP$ is more useful to measure the performance of an algorithm to correctly recommend to users who have long consumption sequences, whereas $MiAP$ is not influenced by the difference of consumption sequence length.

Fig. 5 and Fig. 6 illustrate the $MaAP$ and the $MiAP$ of all the baselines and **TS-PPR** on the two real-world data sets. Obviously, **TS-PPR** outperforms all the baselines on both data sets under most evaluation settings — Top-1, Top-5 and Top-10 recommendations, except for the results of Top-1 recommendation on the Lastfm data set. The relative precision improvement of **TS-PPR** compared with the best of the baselines on the two data sets is shown in Table 3. The accuracy improvement on the Gowalla data set is very significant, especially in the Top-1 recommendation task where the relative improvement is around 80% in both $MaAP$ and $MiAP$. Although the accuracy improvement on the Lastfm data set is less significant compared with that on Gowalla, and our Top-1 recommendation result is even inferior to **Recency** and **Survival** on this data set, **TS-PPR** still outperforms all the baselines in the Top-5 and the Top-10 recommendation tasks. Thus, we can say that **TS-PPR** is effective for the RRC problem in most cases, except that the accuracy performance of the proposed method on the Top-1 recommendation task may fluctuate due to the high volatility in recommending only one item each time. Besides, as shown in Table 3, the improvement of $MaAP@N$ is generally higher than $MiAP@N$ on the Gowalla data set ($N \in \{1, 5, 10\}$). This is because the imbalance of consumption sequence length in Gowalla is more severe than that in Lastfm, and the proposed method is prone to optimize the recommendation results on users with long consumption sequences.

TABLE 3
Relative precision improvement of the proposed method compared with the best of all the baselines. ($\Omega = 10$, $S = 10$)

Data set	$MaAP$			$MiAP$		
	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10
Gowalla	82%	38%	36%	72%	27%	30%
Lastfm	\	3%	9%	\	18%	14%

When looking at the accuracy performance of the baselines in Fig. 5 and Fig. 6, we find that **Pop** is generally more effective than **Random** and **Recency**, which means that item popularity could be more important than recency effect in people's repeat consumption behaviors. We should also note that this conclusion holds when only repeat consumption behaviors from at least Ω steps ago are evaluated. Otherwise, recency effect might be of equal importance as item popularity [7].

As for the baselines of **FPMC**, **Survival** and **DYRC**, we found that **DYRC** [7] generally has higher precision

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

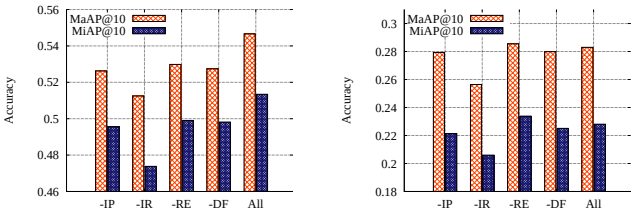
in tackling this RRC problem. It combines item popularity and recency effect in a mixed weighted model. However, it considers item popularity and recency effect as latent weights, and learns these weights through gradient descent, which is different from **TS-PPR** in both feature definition and computational algorithm. **FPMC** [41] shows little difference in the accuracy performance compared with **Pop**, **Random** and **Recency**, which may result from that **FPMC** only considers the transition probability between items using Markov Chain model without using any behavioral features. Furthermore, **Survival** performs poorly on this RRC problem in our experiments. It is because the **Survival** method [30] was defined in the continuous time domain to predict the user return time for a service. However, we are dealing with discrete consumption behaviors in this study. This discretization may greatly decrease the performance of **Survival**.

Through this accuracy comparison, we can see that **TS-PPR** shows the best accuracy performance on real-world data sets in most cases. Meanwhile, it can be seen that the incorporation of time-sensitive behavioral features is very useful to improve the accuracy performance in the RRC problem. However, we can also find the difference in the accuracy improvement of **TS-PPR** on the two data sets. The major reason of this difference could be attributed to the characteristics of behavioral features as illustrated in Fig. 4. **TS-PPR** depends on the behavioral features to train the personalized ranking function and maximize the difference of preference between positive samples and negative samples. Obviously, these generic behavioral features extracted in this work are more representative of people's repeat consumption behaviors in the Gowalla data set compared with that in the Lastfm data set, because the curves on Gowalla in Fig. 4 are generally steeper than those on Lastfm. A steeper curve in this illustration represents a more discriminative feature. Thus, the feature selection has an important influence upon **TS-PPR**. Since only four generic domain-independent behavioral features are extracted in the experiments, we believe that there is still much room to improve the results by incorporating more elaborate off-the-shelf domain-specific features.

5.4 Feature Importance

Although it has been shown that the four features together with **TS-PPR** can lead to superior accuracy in dealing with the RRC problem, it is also interesting to study how each of these features contributes to the accuracy improvement. We conducted experiments to evaluate the importance of these features. In the evaluation, we separately remove one of the four features and leave the rest three to train the recommendation model, and compare the accuracy of the results with that using all four features. Fig. 7 shows the comparison results.

As shown in Fig. 7(a), there is a clear accuracy drop after removing any feature on the Gowalla data set. It can also be seen that there is slight difference in the accuracy when IP, RE or DF is removed. But the accuracy has a much larger drop when removing IR in contrast. This means that IR is a more effective indicator of what a user would like to reconsume. Similar results are also observed on the Lastfm data set in Fig. 7(b), except that the accuracy drop after removing IP, RE or DF is less significant. In sum, the experimental results show that item reconsumption ratio is a better indicator of user repeat consumption behaviors among the four proposed features, but it is still the best to have all the four features work together.



(a) Gowalla (b) Lastfm

Fig. 7. Comparisons of feature importance in the **TS-PPR** model. IP, IR, RE, DF represent item popularity, item reconsumption ratio, recency, dynamic familiarity, respectively. -*X* means the *X* feature is removed. "All" means four features are used.

TABLE 4
Default settings of parameters.

Gowalla	Param	λ	γ	K	S	Ω
	Value	0.01	0.05	40	10	10
Lastfm	Param	λ	γ	K	S	Ω
	Value	0.001	0.1	40	10	10

5.5 Sensitivity of Parameters

Next, since there are multiple important parameters in **TS-PPR**, we evaluate the sensitivity of these parameters in the experiments. The default settings of these parameter values are listed in Table 4.

5.5.1 Regularization Parameters

First, the sensitivity of regularization parameters λ and γ is evaluated. These parameters place penalty on the magnitude of $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{A}$. Fig. 8 shows the performance of **TS-PPR** with the variations in λ and γ values. We can see that the curves of $MaAP@10$ and $MiAP@10$ for both Gowalla and Lastfm data sets are very much alike. There is a significant accuracy drop when the large values of λ and γ underfit the proposed model in the Gowalla data set. Thus, we select $\lambda = 0.01$ and $\gamma = 0.05$ on this data set. While on the Lastfm data set, the influence of λ and γ seems very slight. The optimal setting of λ on this data set is $\lambda = 0.001$, whereas that of $\gamma = 0.1$ to avoid both overfitting and underfitting. Generally, we observe a larger optimal value of γ than that of λ on both data sets. Since γ places penalty on the magnitude of $\mathbf{U}$ and $\mathbf{V}$, while λ on that of $\mathbf{A}$, we can say that the magnitude of $\mathbf{U}$ and $\mathbf{V}$ is more likely to harm the effectiveness of **TS-PPR**.

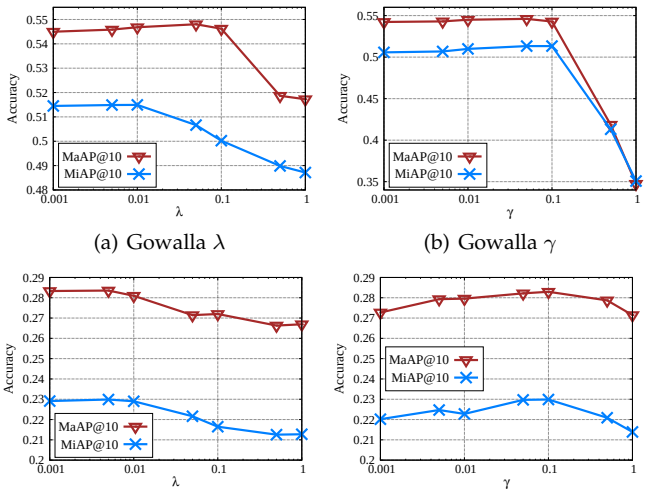
Although it still seems that a smaller λ (< 0.001) may lead to even better performance, the improvement seems very slight according to the trends in Fig. 8(a) and Fig. 8(c). Besides, a smaller λ value increases the possibility of model overfitting.

5.5.2 Latent Feature Space Dimension K

The dimension K of latent feature space has an impact on the computational complexity of **TS-PPR**. The larger the value of K is, the higher computational complexity **TS-PPR** will have. Besides, increasing the latent space dimension

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

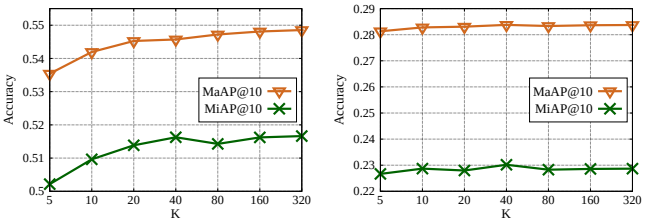
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering



(a) Gowalla λ (b) Gowalla γ

(c) Lastfm λ (d) Lastfm γ

Fig. 8. Influence of regularization parameters λ and γ.



(a) Gowalla K (b) Lastfm K

Fig. 9. Sensitivity of latent feature space dimension K.

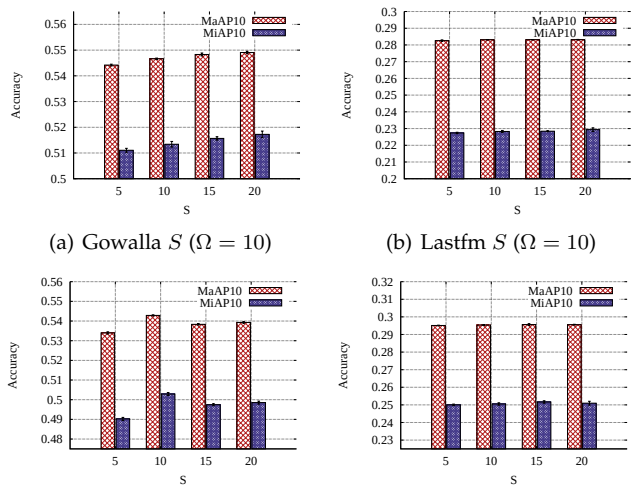
may also increase the expressiveness of **TS-PPR**, and lead to higher prediction accuracy. Fig. 9 illustrates the accuracy ($MaAP@10$ and $MiAP@10$) of **TS-PPR** with variations in K values on the two data sets.

Apparently, the impact of K on the Gowalla data set is more significant than that on the Lastfm data set. There is an increasing trend of accuracy along the larger values of K on Gowalla. This is because higher latent dimensions enable more information to be encoded in the feature vector, and thus the recommendation accuracy can be improved. However, we can also find that the improvement of accuracy gradually converges after $K = 40$. In order to constrain the computational complexity, we choose the default setting as $K = 40$ with near optimal accuracy.

5.5.3 Negative Sample Number S

In this experiment, we change the number S of negative samples w.r.t. a single positive sample in the parameter inference. Since we need to pre-compute the behavioral feature $\mathbf{f}$ for each selected negative sample in advance of training, the value of S is proportional to the computation and storage cost, i.e. the larger the value of S is, the higher the computation and storage cost will be. Fig. 10 shows the recommendation accuracy by varying S values on both data sets. The evaluation with two different settings of the minimum gap Ω ($\Omega = 10, 20$) is presented.

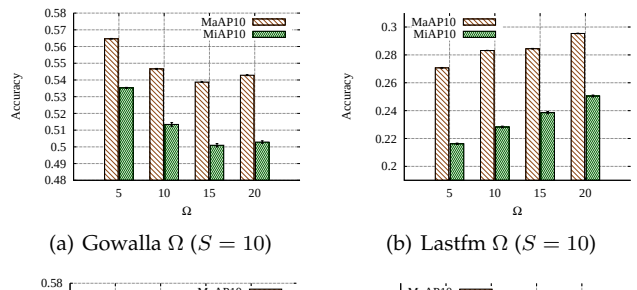
By comparing the results on both data sets, we find that the value of S has less influence on the recommendation accuracy in Lastfm where the values of both $MaAP@10$ and $MiAP@10$ barely change with the variations in S values. In contrast, there is a slight uprising trend in both $MaAP@10$ and $MiAP@10$ with the increase of S on Gowalla (Fig. 10(a) and Fig. 10(c)), except for the small bulge in Fig. 10(c) when $S = 10$, which is most probably due to the parameter tuning on other parameters (e.g. λ, γ, K) towards the optimal accuracy when $S = 10$ as the default. The slight uprising trend of accuracy on Gowalla results from incorporating more training quadruples (u, v_i, v_j, t) which enrich the preference information. However, the increase of S does not change the recommendation accuracy on Lastfm. This is mostly because increasing S does not enlarge the set of different training quadruples (u, v_i, v_j, t) on this data set, which brings little extra preference information to **TS-PPR**.



(a) Gowalla S ($\Omega = 10$) (b) Lastfm S ($\Omega = 10$)

(c) Gowalla S ($\Omega = 20$) (d) Lastfm S ($\Omega = 20$)

Fig. 10. Sensitivity of negative sample number S.



(a) Gowalla Ω ($S = 10$) (b) Lastfm Ω ($S = 10$)

(c) Gowalla Ω ($S = 20$) (d) Lastfm Ω ($S = 20$)

Fig. 11. Sensitivity of the minimum gap Ω.

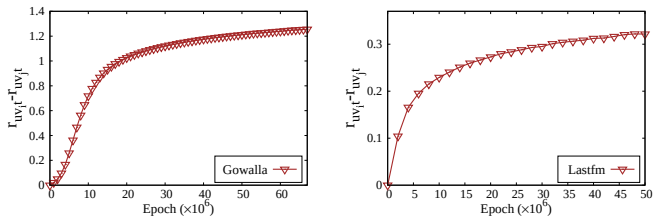
In all, since increasing S does not seem to bring significant improvement on recommendation accuracy on both data sets, we set $S = 10$ as default to save computation and storage cost.

5.5.4 Minimum Gap Ω

In the experiments, we evaluate the performance of **TS-PPR** on repeat consumption behaviors at least Ω steps before the current step. Our major argument is that those items consumed in the last Ω steps have no need to be recommended, because users mostly know how to get the recently consumed items by themselves without recommendation.

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering



(a) Gowalla (b) Lastfm

Fig. 12. Model convergence ($S = 10, \Omega = 10$).

Fig. 11 illustrates the results by varying Ω values on the two data sets. Also, the evaluation results are presented under the settings of both $S = 10$ and $S = 20$. It is easy to notice that the recommendation accuracy on Gowalla is decreasing as Ω gets larger (see Fig. 11(a) and Fig. 11(c)), whereas it is increasing on Lastfm (see Fig. 11(b) and Fig. 11(d)). This means that **TS-PPR** does a better job in recommending recent repeat consumption than recommending remote repeat consumption on Gowalla, while it performs better in recommending remote repeat consumption than recent repeat consumption on Lastfm. Generally, as the value of Ω increases, the size ($|\mathbf{W}| - \Omega$) of the candidate set for recommendation shrinks since $|\mathbf{W}|$ does not change. The increase of Ω is supposed to lead to accuracy improvement as what has been observed on Lastfm. Thus, the reason why a reverse trend is observed on Gowalla should be attributed to the more significant recency effect on Gowalla. This leads to the better accuracy performance in recommending recent repeat consumption. In this paper, the default setting of Ω is 10 unless explicitly stated.

5.6 Efficiency Evaluation

We also evaluate the efficiency performance of **TS-PPR** on the two data sets. The evaluation was conducted on a 64-bit Windows Server 2008 with Intel(R) Xeon(R) CPU E5-2650, 2.0GHz, 256GB RAM.

5.6.1 Convergence

Firstly, we evaluate the convergence of the objective function $\mathcal{J}$ using stochastic gradient descent (with the small batch approach). In **TS-PPR**, we try to maximize the probability of $p(v_i >_{ut} v_j)$ for each training quadruple (u, v_i, v_j, t). Since $p(v_i >_{ut} v_j) = \sigma(r_{uv_it} - r_{uv_jt})$ and σ is a monotonically increasing function, we actually maximize $r_{uv_it} - r_{uv_jt}$ in the parameter inference process. Let $\tilde{r}$ be the average of $r_{uv_it} - r_{uv_jt}$ for all the quadruples in the small batch, and $\Delta\tilde{r}$ be the change of $\tilde{r}$ between adjacent convergence check points. To maximize $r_{uv_it} - r_{uv_jt}$ equals to maximize $\tilde{r}$ in the parameter inference. Thus, the change $\Delta\tilde{r}$ becomes an ideal measurement to check the model convergence. For example, we may consider the proposed model converged in the training when $\Delta\tilde{r} \leq 10^{-3}$. We think that $\Delta\tilde{r}$ is a better choice to check for convergence compared with the average of $\mathcal{J}_{small}$, because the average of $\mathcal{J}_{small}$ is limited between the range $(0, 1)$ and the value change between the adjacent small batches may be very small due to the range limitation. On the contrary, there is no range limitation on $\Delta\tilde{r}$.

Fig. 12 shows the model convergence in the parameter inference on the two data sets. We use the same convergence condition $\Delta\tilde{r} \leq 10^{-3}$ on both data sets. Apparently, the number of epochs between adjacent convergence check points on Lastfm is larger compared with that on Gowalla because it is proportional to the size of the training set $\mathcal{D}$ in our experiments. Due to the same range of x-axis in Fig. 12(a) and 12(b), the points in Fig. 12(b) are sparser. Besides, the maximum value of $\tilde{r}$ on Gowalla after convergence is also higher than that on Lastfm. It means that the positive samples v_i in Gowalla are easier to be distinguished from the negative samples v_j using **TS-PPR**, which is the reason why the accuracy improvement on Gowalla is much more significant compared with that on Lastfm (see Fig. 5 and Fig. 6). Through this comparison, we find that the accuracy performance of **TS-PPR** can be reflected on the average preference difference $\tilde{r}$ obtained in the parameter inference process. The larger the value of $\tilde{r}$ is after parameter inference, the more likely that the accuracy improvement will be higher. This can be used to optimize the parameter inference as well.

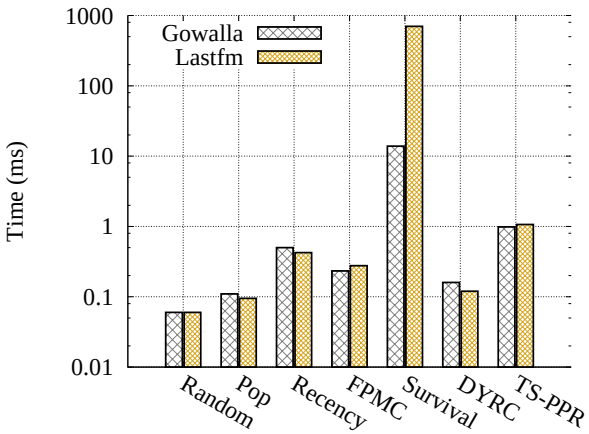


Fig. 13. Average online recommendation time of a single instance measured by milliseconds.

5.6.2 Online Recommendation Time Cost

Fig. 13 shows the average online recommendation time for a single instance measured by milliseconds (in log scale). The data is reported by averaging results on 3 trials each. Basically, **Random**, **Pop** and **DYRC** have a low online recommendation time cost because they only need a one-pass simple multiplication along the time window to generate recommendation. **Recency** has a higher cost due to the exp computation in its weighting scheme. **FPMC** has a medium time cost resulting from the inner product of latent features in the preference computation. **Survival** has the highest time cost in this comparison, which is 2—4 orders of that of **Random**, **Pop** and **DYRC**. This is because **Survival** requires to compute the feature, *time weighted average return time*, online. The computational cost of this feature is proportional to the length of the whole consumption sequence. The online recommendation time cost of **TS-PPR** is higher than other baselines but much lower than **Survival**. However, since the average online time cost of **TS-PPR** for a single instance is only around 1 millisecond which does not affect the user experience, the proposed method is still efficient enough.

As for the offline training time cost, since the training time varies too much and it is not the most essential factor of a recommender system, we report these data without

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

TABLE 5
Evaluation combining **STREC** and **TS-PPR**.

Data Set	**STREC**	**TS-PPR** (on **STREC** correct classification)		
		MaAP@1	MaAP@5	MaAP@10
Gowalla	0.6912	0.1343	0.4487	0.6314
Lastfm	0.8070	0.0862	0.2819	0.4336

illustration. First, there is no training stage for all the simple baselines like **Random**, **Pop** and **Recency**. The training time of the proposed method is subject to the predefined *convergence precision* $\Delta\tilde{r}$. For example, when setting $\Delta\tilde{r} < 10^{-3}$ as the convergence condition, the training time of **TS-PPR** on each data set is around 1—2 hour(s) with a single running thread, which is similar to that of **FPMC** and **Survival** in the parameter inference procedure. Meanwhile, **DYRC** has the highest training cost in the comparison, which takes 2—3 days to train its model on Lastfm even in parallel with 24 running threads.

In all, **TS-PPR** has acceptable training and online recommendation time cost. Considering its dominating recommendation accuracy compared with the baselines, we are convinced that **TS-PPR** is both effective and efficient.

5.7 Combination of TS-PPR with STREC

As pointed out in Section 1, **RRC** addressed in this paper is the consecutive problem of the previous work **STREC** [13] in recommender system research. Thus, we combine **STREC** and **TS-PPR** together as a holistic algorithm for repeat consumption behavior prediction. More specifically, we perform personalized recommendations with **TS-PPR** on real repeat consumption correctly identified by **STREC**. For **STREC**, we choose the linear model in [13]. For **TS-PPR**, we use the default settings in Table 4. The same data sets and data processing are used as with the previous experiments in this paper. The experimental results are shown in Table 5. The accuracy of **STREC** to predict whether or not a user will perform a repeat consumption is about 0.7 and 0.8 on the two data sets, respectively. Meanwhile, the recommendation accuracy of **TS-PPR** conditional on the repeat consumption correctly classified by **STREC** is also very promising (e.g. 0.6314 MaAP@10 on Gowalla). By timing the numbers under **STREC** and **TS-PPR**, we will get the accuracy of jointly addressing both the **RRC** and the **STREC** problems at the same time. For example, the accuracy of combining **STREC** and **TS-PPR** in the problem of classifying (**STREC**) and making recommendation (**RRC**) for potential repeat consumption behaviors on Gowalla data set is around 0.44 (0.6912×0.6314) on MaAP@10. Besides, the accuracy performance of **TS-PPR** alone is shown in Fig. 5 and 6.

6 Conclusion and Future Work

In this paper, we studied the problem of recommendation for repeat consumption to broaden the application areas of recommender systems. We argue that personalized recommendation for repeat consumption also has a real utility just like what have been studied on novel item recommendation. To address this problem, we bring forward a time-sensitive pairwise ranking model to combine users' static and dynamic preferences towards the previously consumed items to perform personalized recommendation. The proposed model factorizes the temporal user-item interactions via learning personalized mappings from behavioral features in observable space to preference features in latent space. We show through experimental evaluation the effectiveness and the efficiency of the proposed method compared with all the baselines.

As for the future work, we will focus on mixing the results of recommendations for both novel consumption and repeat consumption, through which recommender system will be more flexible and user-friendly by providing recommendations with novelty, familiarity and value.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 61373023, No. 61170064), and the National High Technology Research and Development Program of China (No. 2013AA013204).

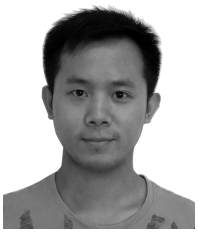
References

[1] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.
[2] N. Koenigstein, G. Dror, and Y. Koren, "Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy," in *RecSys*, 2011, pp. 165–172.
[3] R. M. Bell and Y. Koren, "Lessons from the netflix prize challenge," *ACM SIGKDD Explorations Newsletter - Special issue on visual analytics*, vol. 9, no. 2, pp. 75–79, 2007.
[4] X. Amatriain and B. Mobasher, "The recommender problem revisited: morning tutorial," in *SIGKDD*, 2014, pp. 1971–1971.
[5] N. Hurley and M. Zhang, "Novelty and diversity in top-n recommendation – analysis and evaluation," *ACM Transactions on Internet Technology*, vol. 10, no. 14, March 2011.
[6] Y. C. Zhang, D. O. Séaghdha, D. Quercia, and T. Jambor, "Auralist: introducing serendipity into music recommendation," in *WSDM*, 2012, pp. 13–22.
[7] A. Anderson, R. Kumar, A. Tomkins, and S. Vassilvitskii, "The dynamics of repeat consumption," in *WWW*, 2014, pp. 419–430.
[8] F. Zhang, K. Zheng, N. J. Yuan, X. Xie, E. Chen, and X. Zhou, "A novelty-seeking based dining recommender system," in *WWW*, 2015, pp. 1362–1372.
[9] K. Kapoor, K. Subbian, J. Srivastava, and P. Schrater, "Just in time recommendations: Modeling the dynamics of boredom in activity streams," in *WSDM*, 2015, pp. 233–242.
[10] H. Ebbinghaus, *Memory; A Contribution to Experimental Psychology*. New York by Teachers College, Columbia University, 1885.
[11] M. Y. Jaber and M. Bonney, "A comparative study of learning curves with forgetting," *Applied Mathematical Modelling*, vol. 21, no. 8, pp. 523–531, 1997.
[12] L. Averell and A. Heathcote, "The form of the forgetting curve and the fate of memories," *Journal of Mathematical Psychology*, vol. 55, pp. 25–35, 2011.
[13] J. Chen, C. Wang, and J. Wang, "Will you "reconsume" the near past? fast prediction on short-term reconsumption behaviors," in *AAAI Conference on Artificial Intelligence*, 2015, pp. 23–29.
[14] J. Chen, C. Wang, and J. Wang, "A personalized interest-forgetting markov model for recommendations," in *AAAI Conference on Artificial Intelligence*, 2015, pp. 16–22.
[15] J. Chen, C. Wang, and J. Wang, "Modeling the interest-forgetting curve for music recommendation," in *ACM Multimedia*, 2014, pp. 921–924.
[16] L. D. Catledge and J. E. Pitkow, "Characterizing browsing strategies in the world-wide web," *Computer Networks and ISDN Systems*, vol. 27, no. 6, pp. 1065–1073, 1995.
[17] E. Adar, J. Teevan, and S. T. Dumais, "Large scale analysis of web revisitation patterns," in *CHI*, 2008, pp. 1197–1206.
[18] H. Zhang and S. Zhao, "Measuring web page revisitation in tabbed browsing," in *CHI*, 2011, pp. 1831–1834.

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2016.2593720, IEEE Transactions on Knowledge and Data Engineering

[19] J. Liu, C. Yu, W. Xu, and Y. Shi, "Clustering web pages to facilitate revisitation on mobile devices," in *IUI*, 2012, pp. 249–252.
[20] R. Kawase, G. Papadakis, and E. Herder, "Supporting revisitation with contextual suggestions," in *ACM/IEEE joint conference on Digital libraries*, 2011, pp. 227–230.
[21] R. Kawase, E. Herder, and W. Nejdl, "Beyond the usual suspects: context-aware revisitation support," in *ACM conference on Hypertext and hypermedia*, 2011, pp. 27–36.
[22] C. Yu, R. Balakrishnan, K. Hinckley, T. Moscovich, and Y. Shi, "Implicit bookmarking: Improving support for revisitation in within-document reading tasks," *International Journal of Human-Computer Studies*, vol. 71, pp. 303–320, 2013.
[23] S. K. Tyler and J. Teevan, "Large scale query log analysis of re-finding," in *WSDM*, 2010, pp. 191–200.
[24] R. Jones and K. L. Klinkner, "Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs," in *CIKM*, 2008, pp. 699–708.
[25] J. Teevan, E. Adar, R. Jones, and M. Potts, "Information re-retrieval: repeat queries in yahoo's logs," in *SIGIR*, 2007, pp. 151–158.
[26] D. Elsweiler, M. Baillie, and I. Ruthven, "What makes re-finding information difficult? a study of email re-finding," in *ECIR*, 2011, pp. 568–579.
[27] R. Kawase, G. Papadakis, E. Herder, and W. Nejdl, "The impact of bookmarks and annotations on refinding information," in *ACM conference on Hypertext and hypermedia*, 2010, pp. 29–34.
[28] T. Deng, L. Zhao, L. Feng, and W. Xue, "Information re-finding by context: a brain memory inspired approach," in *CIKM*, 2011, pp. 1553–1558.
[29] T. Deng, L. Zhao, H. Wang, Q. Liu, and L. Feng, "Refinder: A context-based information refinding system," *TKDE*, vol. 25, no. 9, pp. 2119–2132, 2013.
[30] K. Kapoor, M. Sun, J. Srivastava, and T. Ye, "A hazard based approach to user return time prediction," in *SIGKDD*, 2014, pp. 1719–1728.
[31] K. Kapoor, N. Srivastava, J. Srivastava, and P. Schrater, "Measuring spontaneous devaluations in user preferences," in *SIGKDD*, 2013, pp. 1061–1069.
[32] D. Lian, X. Xie, V. W. Zheng, N. Yuan, and F. Zhang, "CEPR: A collaborative exploration and periodically returning model for location prediction," *ACM Transactions on Intelligent Systems and Technology*, vol. 6, no. 1, April 2015.
[33] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.
[34] T. V. Nguyen, A. Karatzoglou, and L. Baltrunas, "Gaussian process factorization machines for context-aware recommendations," in *SIGIR*, 2014, pp. 63–72.
[35] Y. Yao, H. Tong, G. Yan, F. Xu, X. Zhang, B. K. Szymanski, and J. Lu, "Dual-regularized one-class collaborative filtering," in *CIKM*, 2014, pp. 759–768.
[36] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *ACM WSDM*, 2014, pp. 273–282.
[37] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *ICDM*, 2008, pp. 263–272.
[38] V. C. Ostuni, T. D. Noia, E. D. Sciascio, and R. Mirizzi, "Top-n recommendations from implicit feedback leveraging linked open data," in *RecSys*, 2013, pp. 85–92.
[39] U. Paquet and N. Koenigstein, "One-class collaborative filtering with random graphs," in *WWW*, 2013, pp. 999–1008.
[40] R. Pan, Y. Zhou, B. Cao, and N. Liu, "One-class collaborative filtering," in *ICDM*, 2008, pp. 502–511.
[41] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*, 2010, pp. 811–820.
[42] D. Rafailidis and A. Nanopoulos, "Modeling the dynamics of user preferences in coupled tensor factorization," in *RecSys*, 2014, pp. 321–324.
[43] D. Rafailidis and A. Nanopoulos, "Repeat consumption recommendation based on users preference dynamics and side information," in *WWW Companion*, 2015, pp. 99–100.
[44] D. M. Dunlavy, T. G. Kolda, and E. Acar, "Temporal link prediction using matrix and tensor factorizations," *TKDD*, vol. 5, no. 2, February 2011.
[45] C. Zhang, K. Wang, H. Yu, J. Sun, and E.-P. Lim, "Latent factor transition for dynamic collaborative filtering," in *SDM*, 2014, pp. 452–460.
[46] H. Gao, J. Tang, X. Hu, , and H. Liu, "Exploring temporal effects for location recommendation on location-based social networks," in *RecSys*, 2013, pp. 93–100.
[47] Y. Wang, N. J. Yuan, D. Lian, L. Xu, X. Xie, E. Chen, and Y. Rui, "Regularity and conformity: Location prediction using heterogeneous mobility data," in *SIGKDD*, 2015, pp. 1275–1284.
[48] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *IJCAI*, 2013, pp. 2605–2611.
[49] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *SIGKDD*, 2011, pp. 1082–1090.
[50] O. Celma, *Music Recommendation and Discovery in the Long Tail*. Springer, 2010, ch. 3.



Jun Chen received the B.S. degree in Computer Software from Tsinghua University, Beijing, China, in 2012. From 2012 to 2014, he studied at School of Software, Tsinghua University, as a Master student before his enrollment of PhD program. He is currently a PhD candidate majoring in Software Engineering at School of Software, Tsinghua University. His research interests include personalization and recommender systems, user behavior analysis and prediction, and general topics in machine learning. He has published research papers in major conferences and journal including *AAAI*, *ACM Multimedia*, *CIKM*, and *IEEE Signal Processing Letters*.

Chaokun Wang received the B.Eng. degree in computer science and technology, the M.Sc. degree in computational mathematics and the Ph.D. degree in computer software and theory in 1997, 2000 and 2005, respectively, from Harbin Institute of Technology, China. He joined the faculty of School of Software at Tsinghua University in February 2006, where currently he is an Associate Professor. He has published over 60 refereed papers, got two best paper awards at international conferences and holds twelve patents. His current research interests include social network analysis, graph data management, and music computing.

Jianmin Wang graduated from Peking University, Beijing, China, in 1990, and received the M.E. and the Ph.D. degrees in computer software from Tsinghua University, China, in 1992 and 1995, respectively. He is a full professor in the School of Software, Tsinghua University. His research interests include unstructured big data management, workflow and BPM technology, and large-scale data analytics. He has published 100 papers in major journals and conferences, including *TKDE*, *DMKD*, *WWWJ*, *SIGMOD*, *VLDB*, *ICDE*, *SIGKDD*, *SIGIR*, *AAAI*, *CVPR*, and *ICCV*. He led to develop a product data/lifecycle management system, which has been implemented in hundreds of enterprises in China. He leads to develop an unstructured big data management system, LaUDMS.

Philip S. Yu received the PhD degree in EE from Stanford University. He is a distinguished professor in computer science at University of Illinois at Chicago and is the Wexler chair in Information Technology. His research interests include data mining, data stream, database and privacy. He is the editor-in-chief of the *ACM Transactions on Knowledge Discovery from Data*. He was the editor-in-chief of the *IEEE Transactions on Knowledge and Data Engineering* (2001-2004). He received a Research Contributions Award from *IEEE International Conference on Data Mining* (2003) and a Technical Achievement Award from *IEEE Computer Society* (2013). He is a fellow of the IEEE.

1041-4347 (c) 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.