SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2019, Vol. 62 212105:1–212105:14
https://doi.org/10.1007/S11432-018-9915-8

Recommendation over time: a probabilistic model of time-aware recommender systems

Zuoquan LIN* & Hanxuan CHEN

Department of Information Science, School of Mathematical Sciences, Peking University, Beijing 100871, *China*

Received 31 December 2018/Revised 4 March 2019/Accepted 17 May 2019/Published online 9 October 2019

Abstract In time-aware recommender systems, we have to consider the dynamic aspect of recommendation that is fond of new coming data. Usually, the recent data is more closely related to current recommendation tasks and the early data are useful to indicate overall measurements of the preferences. We propose a probabilistic model that uses the early data to generate the prior distribution and the recent data to capture the change of the states of both users and items in collaborative filtering systems. Our model is dynamic in the sense that it updates every time receiving new data. The time cost of every updating has a constant limit, which is suitable to deal with large scale data for online recommendation. Experiments on real datasets show the improvement performance of our model over the existing time-aware recommender systems.

Keywords recommender system, collaborative filtering, hidden Markov model, precision, cold start

Citation Lin Z Q, Chen H X. Recommendation over time: a probabilistic model of time-aware recommender systems. Sci China Inf Sci, 2019, 62(11): 212105, https://doi.org/10.1007/S11432-018-9915-8

1 Introduction

Recommender systems give personalized suggestions to users from a large number of items. The users and items of recommender systems change over time. Any user can go over to different preferences as time goes by. The categories of items are changeable along with the trend of consumption. New users and items are constantly incorporated, which might influence existing ones. Recommender systems have to model the dynamics of recommendation over time.

Considering the behaviors of users, it is useful to define the distinction between the recent interests and the early ones. Recommender systems more concern about "what might the users like now", though they are more or less influenced by "what did the users like before". On the other side, the behaviors of users change with the change of items over time. The distinction between the recent and early data is relative. The early data can be increasingly accumulated by the recent data that is discarded by the newest data in certain time widows. In collaborative filtering systems, the recent ratings, which are closer to current status of both users and items, need to pay more attention. The early ratings are needed and accumulated, which approximate overall measurements of the preferences of both users and items.

Most of time-aware recommender systems use the data of users and items without the distinction between the recent and early data. The time related functions and features are common ways in time-aware recommender systems [1–8]. These methods try to capture temporal properties of all the past time by day or week specific parameters. However, they cannot build up the connection between the recent and early time. Tensor factorizations handle time as a separate dimension [9–11]. By the three-vector inner-product formula, the additional time vector can adjust the weight of each hidden factor. However, they

* Corresponding author (email: linzuoquan@pku.edu.cn)

cannot indicate individual change of each user or item. Similar to the time-related functions, they cannot derive the connection between the recent and early time. The sequence pattern mining in recommender systems is used to learn the behaviors of users [12–19]. The method takes care about the sequence of actions of users in period. However, it does not concern about the difference of the recent and early data and the change of the preferences of users.

There are probabilistic models such as hidden Markov models (HMMs) [20–27] and Kalman filter [28–32] in recommender systems. These studies explain the behaviors of users in probabilistic models for making prediction. When learning the model parameters, they use historical data with the same measurements of the preferences. Unfortunately, the model parameters would be influenced more by the early data rather than the recent data. They cannot well deal with the difference of the recent and early data.

In this paper, we consider the recommendation in collaborative filtering that changes over time by making the distinction between the recent and early data. It updates when it receives the new coming data and accumulates the historical data to approximate overall measurements of the preferences for both interests of users and properties of items. We propose a probabilistic model based on HMM that captures the change of the preferences of both users and items. Moreover, most of existing time-aware recommender systems are offline recommendation by batch learning. They have to retrain the whole model when new data is imported. In our model, the time cost of every update has a constant limit for the recent and early data, which makes it suitable to deal with large scale data for online recommendation. Experiments on real datasets show the improvement performance of our model over the existing time-related recommender systems.

In Section 2, we discuss the related work. We present the probabilistic model for recommendation over time in Section 3. In Section 4, we provide the algorithms of our model. In Section 5, we demonstrate the improvement performance of our model. Finally, we make remarks in the concluding section.

2 Related work

The recommender systems based on neighborhood consider the importance of the recent data. In [33–38], time weights are used as a simple way for time information. They calculate the similarity of items (or users) by increasing the weights of items rated at recent time and make recommendation by similar items. The time window is another way for time information [39, 40]. Only the ratings in a recent time window are considered for the calculation of similar items and the others are discarded. However, the time weights and the time window are so simple that they cannot distinct the recent and early ratings. In our model, we only store the recent ratings, which is somehow similar to the time window method. We do not discard the early ratings but accumulate them to get the prior distribution of random variables of each user and item. Our model avoids the loss of information in the process of discarding ratings.

In [41, 42], the models use the long-term and short-term preferences to describe the behaviors of users, which is similar to the distinction between the recent and early data in our model. In [43–47], the models concentrate on the short-term interests. Those studies mostly use graph-based algorithms, which are totally different from probabilistic model. The user-item-session graph or user-item networks have some connections about the users and their long-term or short-term preferred items. However, they cannot model the change of the preferences of users by the graphs. Our model represents both the interests of users and properties of items by the random variables with clear interpretations and the change of the preferences of both users and items are represented by the temporal variables as well.

In [20, 24, 25], the hidden (semi-)Markov models are applied for recommender systems, in which the hidden variables are used to describe the states of users and the item selections are the observed evidences. It is a convenient way to use HMMs to describe the item selections of users. In [23, 26], the HMMs are applied to the sequence pattern mining in recommender systems. There are some specific applications of HMMs for sport videos [21] and people-to-people recommendation [22, 27]. However, those models do not have hidden variables for items and thus there are not proper ways to describe the hidden properties

of items and their change over time.

Kalman filter is another probabilistic model used in recommender systems. In [32,48], the models CKF and SRec have the temporal variables similar to our model. The most important difference between our model and those models is that those models do not consider the distinction between the recent and early data. The updating steps of those models use only the most recent rating or ratings of the most recent time point. We set the prior distribution by accumulating the early ratings and run the forward-backward algorithm on the recent ratings. Although only using the most recent rating is faster, our algorithm has better recommendation performance and reasonable constant time cost. In [30, 31], Kalman smoothing is used for all the historical data. As indicated in [32], it is inefficient and not able to handle large scale data. In our model, the time-consuming forward-backward algorithm only runs on the recent ratings and the maximal number of recent ratings is adjustable. Hence, our model is fast and suitable for large scale online recommendation.

In recent years, deep learning is applied for recommender systems. The recurrent neural networks (RNNs) can model long and short term sequential recommendation [49–53]. Though those studies have the inspiration of deep learning in good trade-off between long term and short term prediction, they cannot model the preferences of users and the distinction between the recent and early data. Our work indicates that the probabilistic model gains a competitive advantage over rival deep learning in time-aware recommender systems.

3 Model

3.1 Notations

Let $\mathrm{CF} = \langle \mathrm{User}, \mathrm{Item}, \mathrm{Time}, \mathrm{Level}, \mathrm{Rating} \rangle$ be a collaborative filtering system, where

(1) User is a set of users, and user (or u) $\in$ User is a user.

(2) Item is a set of items, and item (or i) $\in$ Item is an item.

(3) Time $= \mathbb{R}^+$ indicates time, where $\mathbb{R}^+$ is the set of positive real numbers.

(4) Level$= \{1, 2, \ldots, N\}$ is the values of ratings, given an integer N.

(5) Rating is a set of ratings, and rating = (user, item, time, level) (or $r = (u, i, t, l)$) $\in$ Rating is a rating, which means that user gives rating level to item at time.

We write RT(user) and RT(item) as the sets of ratings belonging to a user and an item, respectively. For any rating set Ω, we write subscript to indicate the subset of ratings at a given time. For instance, $\Omega_{t_1} = \{(u, i, t, l) \in \Omega \mid t = t_1\}$. We write $\mathrm{TM}(\Omega)$ to represent the set of times (time points) that there are ratings in Ω, i.e., $\mathrm{TM}(\Omega) = \{t \mid (u, i, t, l) \in \Omega\}$. For a time $t \in \mathrm{TM}(\Omega)$, $t + \delta(\Omega, t)$ means the next time in $\mathrm{TM}(\Omega)$, i.e., $t + \delta(\Omega, t) = \min\{t' \mid t' \in \mathrm{TM}(\Omega) \wedge (t' > t)\}$, and we simply write $t + \delta$ to represent the next time of t in $\mathrm{TM}(\Omega)$ for given Ω.

We shall use the rating set in probabilistic formulas. For instance, $P(\Omega)$ denotes the probability of the event defined by Ω, where each $(u, i, t, l) \in \Omega$ means that the rating, a user u gives to an item i at time t, is level l.

3.2 Random variables

In CF, we collect the records of similar users (or items) and analyze common behaviors among them. We use the hidden (random) variables to indicate the user types that describe the interests of users. The users with the same type have similar opinion to the items. Any user has a type at specific time. Because the interests of a user change over time, the user type changes.

We define the sets of user types $\mathrm{UserType} = \{1, \ldots, J\}$ and item types $\mathrm{ItemType} = \{1, \ldots, K\}$, where J and K are integers, respectively. For each user, we define temporal variables $X_{\mathrm{user},t} \in \mathrm{UserType}$ and $Y_{\mathrm{item},t} \in \mathrm{ItemType}$ to describe which type the user and item belong to at time t, respectively. We denote the starting time of $X_{\mathrm{user},t}$ and $Y_{\mathrm{item},t}$ as τ_{user} and τ_{item}, respectively. For rating, we define the

variable $R_{\text{user,item},t} \in$ Level for each triplet $(\text{user}, \text{item}, t)$. In CF, by some known ratings, $R_{\text{user,item},t}$ are (partially) observed, while $X_{\text{user},t}$ and $Y_{\text{item},t}$ are hidden states.

For example, User $= \{u_1, u_2\}$, Item $= \{i_1, i_2, i_3\}$, $N = 5$, Rating $= \{(u_1, i_1, 1, 3), (u_2, i_2, 2, 5), (u_1, i_2, 4, 4), (u_2, i_3, 6, 2)\}$ and $J = 3$, $K = 2$. There are two user variables $X_{u_1,t}$ and $X_{u_2,t}$ with the values in UserType $= \{1, 2, 3\}$. There are three item variables $Y_{i_1,t}$, $Y_{i_2,t}$ and $Y_{i_3,t}$ with the values in ItemType $= \{1, 2\}$. There are four observed rating variables $R_{u_1,i_1,1} = 3$, $R_{u_2,i_2,2} = 5$, $R_{u_1,i_2,4} = 4$, and $R_{u_2,i_3,6} = 2$.

As usually, we assume that the variables for different users and items are independent for the sake of simplicity. For the change of the variables over time, we assume that $X_{\text{user},t}$ follows from the continuous-time Markov assumptions: $\forall t \geqslant 0$, $\tau_{\text{user}} \leqslant s_0 < \cdots < s_n < s$ and $1 \leqslant i, j, i_0, \ldots, i_n \leqslant J$,

$$\begin{aligned} &P(X_{\text{user},s+t} = j \mid X_{\text{user},s} = i, X_{\text{user},s_n} = i_n, \ldots, X_{\text{user},s_0} = i_0) \\ &\quad = P(X_{\text{user},\tau_{\text{user}}+t} = j \mid X_{\text{user},\tau_{\text{user}}} = i). \end{aligned} \tag{1}$$

Thus, $X_{\text{user},t}$ consists of a continuous-time Markov chain. For any t, there is a $J \times J$ transition matrix P_t describing the change of random variable:

$$P(X_{\text{user},s+t} = j \mid X_{\text{user},s} = i) = (P_t)_{i,j}. \tag{2}$$

By Kolmogorov equations [54], a continuous-time Markov chain is described by the jump rate matrix Q. For any t, we have the following equations, where exp and ln are the matrix exponential and the matrix logarithm, respectively:

$$Q = \frac{1}{t}\ln(P_t), \tag{3}$$

$$P_t = \exp(tQ). \tag{4}$$

We write a parameter matrix $A = Q$, which is shared by all the users to describe $X_{\text{user},t}$.

Symmetrically, $Y_{\text{item},t}$ follows from the continuous-time Markov assumptions and consists of a continuous-time Markov chain. We write a parameter matrix B the same as the jump rate matrix, which is shared by all the items.

We consider the relationship among the variables $X_{\text{user},t}$, $Y_{\text{item},t}$ and $R_{\text{user,item},t}$. When a user of the j-th type meets an item of the k-th type, the user prefers the item with the probability $p_{j,k}$. We use the binomial distribution $B(N-1, p_{j,k})$ to convert $p_{j,k}$ into discrete ratings:

$$\begin{aligned} &P(R_{\text{user,item},t} = l \mid X_{\text{user},t} = j, Y_{\text{item},t} = k) \\ &\quad = \Pr(l-1; N-1, p_{j,k}) \\ &\quad = \binom{N-1}{l-1}(p_{j,k})^{l-1}(1-p_{j,k})^{N-l}. \end{aligned} \tag{5}$$

3.3 Early ratings

We divide the ratings of a(n) user (item) into the early ones and the recent ones. The latest ratings of a(n) user (item) are called the recent ratings, and the others called the early ratings. We denote the sets of the early ratings and the recent ratings for user as ER(user) and RR(user), respectively. We have the following conditions:

$$\begin{aligned} \text{RT(user)} &= \text{ER(user)} \cup \text{RR(user)}, \\ \text{ER(user)} &\cap \text{RR(user)} = \emptyset. \end{aligned} \tag{6}$$

Similarly, the sets of the early ratings and the recent ratings for item are denoted as ER(item) and RR(item) with the similar conditions, respectively.

By the partition of the ratings of user, let $X_{\text{user},t}$ start at the first recent rating time τ_{user}. The early ratings of user are used to generate the prior distribution of the random variable, i.e., $P(X_{\text{user},\tau_{\text{user}}} = j)$. The recent ratings of user are used to analyze the change of the interests of the user in recent time.

We consider the generation of the prior distribution of the variable in the following:

$$P(X_{\text{user},\tau_{\text{user}}}=j)=\frac{\pi_j\phi_{\text{user},j}}{\sum_{k=1}^{J}\pi_k\phi_{\text{user},k}}. \tag{7}$$

The first part π_j is a parameter shared by all the users. It is about the global probability of the users belonging to each type. The second part $\phi_{\text{user},j}$ is about the probability of the users belonging to each type on the basis of the previous behaviors of the user, which is a local value generated by the early ratings in

$$\phi_{\text{user},j}=\prod_{(u,i,t,l)\in \text{ER(user)}}\sum_{k=1}^{K}q(\text{item},t,k)\Pr(l-1;N-1;p_{j,k}), \tag{8}$$

where $q(\text{item},t,k)$ is defined as follows:

$$q(\text{item},t,k)=\begin{cases}P(Y_{\text{item},t}=k), & t\geqslant\tau_{\text{item}},\\ g_{\text{item},t,k}, & t<\tau_{\text{item}}.\end{cases} \tag{9}$$

In (8), for $t\geqslant\tau_{\text{item}}$, we have the following equation according to (5):

$$P(R_{\text{user,item},t}=l\mid X_{\text{user},t}=j)=\sum_{k=1}^{K}P(Y_{\text{item},t}=k)\Pr(l-1;N-1;p_{j,k}). \tag{10}$$

Note that if $t<\tau_{\text{item}}$, $P(Y_{\text{item},t}=k)$ is illegal because the starting time of Y_{item} set at τ_{item}. In this situation, we use a parameter $g_{\text{item},t,k}$ to approximate it.

3.4 Recent ratings

By the prior distribution of the early ratings, we can use the recent ratings of a(n) user (item) to infer the change of the user (item) types over time. The goal is to figure out $\gamma_{t,j}=P(X_{\text{user},t}=j\mid \text{RR(user)})$ and $\xi_{t,i,j}=P(X_{\text{user},t}=i,X_{\text{user},t+\delta}=j\mid \text{RR(user)})$. These posterior distributions explain the user types after knowing the ratings and their change.

Firstly, we need to know the probability that the user generates the ratings with respect to each type. For $t\in\text{TM(RR(user))}$, we define $E_{\text{user},t,j}$ as the probability of that user gives all ratings just the same as the elements of RR(user)_t with respect to the j-th type:

$$E_{\text{user},t,j}=P(\text{RR(user)}_t\mid X_{\text{user},t}=j). \tag{11}$$

Assume that the ratings from the users are independent to each other if their types are given, $E_{\text{user},t,j}$ can be calculated by (5) as follows:

$$E_{\text{user},t,j}=\prod_{(u,i,t,l)\in \text{RR(user)}}\sum_{k=1}^{K}q(\text{item},t,k)\Pr(l-1;N-1,p_{j,k}), \tag{12}$$

where we use $q(\text{item},t,k)$ instead of $P(Y_{\text{item},t}=k)$ for the rating (u,i,t,l) is an early rating for item.

Then, we calculate the posterior distributions. To compute γ and ξ, we need $\alpha_{t,j}=P(X_{\text{user},t}=j,\text{RR(user)}_{[\tau_{\text{user}},t]})$ and $\beta_{t,j}=P(\text{RR(user)}_{(t,T]}\mid X_{\text{user},t}=j)$. In fact, we only need to compute these values for $t\in\text{TM(RR(user))}$. We have

$$\alpha_{\tau_{\text{user}},j}=\pi_j\phi_{\text{user},j}E_{\text{user},\tau_{\text{user}},j}, \tag{13}$$

$$\alpha_{t+\delta,j}=E_{\text{user},t+\delta,j}\sum_{i=1}^{J}\alpha_{t,i}\exp(\delta Q)_{i,j}, \tag{14}$$

$$\beta_{T,j}=1, \tag{15}$$

$$\beta_{t,i}=\sum_{j=1}^{J}\beta_{t+\delta,j}\exp(\delta Q)_{i,j}E_{\text{user},t+\delta,j}, \tag{16}$$

$$\gamma_{t,i} = \frac{\alpha_{t,i}\beta_{t,i}}{\sum_{j=1}^{J}\alpha_{t,j}\beta_{t,j}}, \tag{17}$$

$$\xi_{t,i,j} = \frac{\alpha_{t,i}\exp(\delta Q)_{i,j}\beta_{t+\delta,j}E_{\text{user},t+\delta,j}}{\sum_{k=1}^{J}\sum_{m=1}^{J}\alpha_{t,k}\exp(\delta Q)_{k,m}\beta_{t+\delta,m}E_{\text{user},t+\delta,m}}. \tag{18}$$

The computation of these values calls the Forward-Backward algorithm [55]. The jump rate matrix Q is stored in parameters A and B for users and items, respectively. First, we initialize $\alpha_{\tau_{\text{user}},j}$ by (13), and use (14) to compute forward all $\alpha_{t,j}$. Then, we initialize $\beta_{T,j}$ by (15), and use (16) to compute backward all $\beta_{t,i}$. Finally, we compute $\gamma_{t,i}$ and $\xi_{t,i,j}$ by (17) and (18), respectively.

3.5 Parameter estimation

By computing the posterior distributions about user, we update the global parameters π_i, $A_{i,j}$ and $p_{j,k}$. Because these parameters are shared by all the users, we need the posterior distributions from them. We use $\gamma(\text{user})$ to represent the γ value from user and $\xi(\text{user})$ in a similar way. Because $p_{j,k}$ are shared by both users and items, we also need $\gamma(\text{item})$ whose computation is symmetrical to $\gamma(\text{user})$.

The global prior π is about the chance that the users belong to each type in the beginning. It is estimated by the percentage of the users belonging to each type at τ_{user}. λ_1 is a given regularization hyperparameter in

$$\pi_i = \frac{\sum_{\text{user}\in\text{User}}\gamma(\text{user})_{\tau_{\text{user}},i} + \lambda_1}{\sum_{\text{user}\in\text{User}}\sum_{j=1}^{J}\gamma(\text{user})_{\tau_{\text{user}},j} + J\lambda_1}. \tag{19}$$

The elements of jump rate matrix $A_{i,j}$ is the probability of that the user changes from type i to type j. Before we compute A, we compute $\Xi(\text{user})$, the averaged and regularized transition matrix for each user in

$$\Xi_{i,j}(\text{user}) = \lambda_2\frac{\sum_{t\in\text{TM}(\text{RR}(\text{user}))}\xi(\text{user})_{t,i,j}\delta(\text{RR}(\text{user}),t)}{\sum_{k=1}^{J}\sum_{t\in\text{TM}(\text{RR}(\text{user}))}\xi(\text{user})_{t,i,k}\delta(\text{RR}(\text{user}),t)} + (1-\lambda_2)\mathbf{1}_{i=j}. \tag{20}$$

The first part of the formula in (20) is an average of the ξ value of the user with $\delta(\text{RR}(\text{user}),t)$ as the weight. The division ensures the sum of each row of the matrix is 1. The second part is for the assumption that makes the users' type stable. The situation that the users' type does not change (i.e., $i=j$) has higher chance. λ_2 is a given hyperparameter. $\mathbf{1}_{i=j}$ is a function that is 1 for $i=j$ and 0 for else.

We use the sum of the matrix logarithm of Ξ from all the users to generate A by (21), where the denominator (the total number of time points that the user has ratings) is an empirically scaling factor in

$$A = \frac{\sum_{\text{user}\in\text{User}}\ln(\Xi(\text{user}))}{\sum_{\text{user}\in\text{User}}|\text{TM}(\text{RR}(\text{user}))|}. \tag{21}$$

The parameter $p_{j,k}$ is the probability of the users with type j and the items with type k. It is estimated by the expected rating as

$$p_{j,k} = \frac{\sum_{(u,i,t,n)\in\text{Rating}}(l-1)\gamma(\text{user})_{t,j}\gamma(\text{item})_{t,k}}{\sum_{(u,i,t,n)\in\text{Rating}}(N-1)\gamma(\text{user})_{t,j}\gamma(\text{item})_{t,k}}. \tag{22}$$

The sum of $\gamma(\text{user})_{t,j}\gamma(\text{item})_{t,k}$ gives the total chance that the user belongs to the j-th type and the item belongs to the k-th type. The $(l-1)/(N-1)$ follows the way to estimate parameters for the binomial distributions.

4 Algorithm

First of all, we illustrate a summary of the hyperparameters, the random variables and the parameters in our model in Table 1.

Table 1 Hyperparameters, random variables and parameters

Symbol	Meaning
J	Number of user types
K	Number of item types
M	Max number of recent ratings
λ_1	Regularization for π and ω
λ_2	Regularization for A and B
$X_{\text{user},t}$	The type user belongs to at time t
$Y_{\text{item},t}$	The type item belongs to at time t
$R_{\text{user,item},t}$	The rating that user give to item at time t
$p_{j,k}$	Probability that type j user likes type k item
$A_{i,j}$	Jump rate matrix for users
$B_{k,m}$	Jump rate matrix for items
π_j	Global prior distribution for users
ω_k	Global prior distribution for items
$f_{\text{user},t,j}$	Approximation of user variable distribution
$g_{\text{item},t,k}$	Approximation of item variable distribution

Table 2 Additional persistent variables

Variable name	Data type	Meaning
τ_{user}	Number	Time of the first recent rating of user
τ_{item}	Number	Time of the first recent rating of item
$\phi_{\text{user},j}$	Number	Local prior for users
$\psi_{\text{item},k}$	Number	Local prior for items
$x_{\text{user},t,j}$	Number	$P(X_{\text{user},t}=j\|\text{Rating})$
$y_{\text{item},t,k}$	Number	$P(Y_{\text{item},t}=k\|\text{Rating})$
RecRating(user)	Queue	Recent rating for user
RecRating(item)	Queue	Recent rating for item

Our algorithms take all the hyperparameters and the parameters as persistent variables, in addition to the variables in Table 2. J, K, M are integers. p is a $J \times K$ matrix and its elements are randomly initialized in $(0, 1)$. A and B are $J \times J$ and $K \times K$ matrixes, respectively, and are initialized as the matrix logarithm of an identity matrix with small noise. π and ϕ (ω and ψ) are vectors with length $J(K)$, with elements initialized to 1. The queues are empty in initial.

4.1 Updating

The updating algorithm is used every time the system receives a new rating $(\text{user}, \text{item}, T, l)$. First, the new rating is pushed into the queues of users and items. Second, we turn some of the recent ratings of the users into the early ratings and store their information in the local prior distribution $\phi_{\text{user},j}$. Third, we calculate the posterior distributions for the variables of the users and update the global parameters. Similarly, we do the same thing for the related items.

The procedures are described for users in detail by the functions ConvertRating and UpdateParameter in Algorithm 1. The procedures for items are analogous.

We describe the algorithm of converting the recent ratings to the early ones. Assume that one of the recent ratings $(\text{user}, \text{item}_1, t_1, l_1)$ is converted into an early one, we update $\phi_{\text{user},j}$ according to (8) in the following:

$$\phi_{\text{user},j} \leftarrow \phi_{\text{user},j} \sum_{k=1}^{K} \hat{q}(\text{item}_1, t_1, k) \Pr(l_1 - 1; N - 1; p_{j,k}), \quad j = 1, 2, \ldots, J, \tag{23}$$

where

$$\hat{q}(\text{item}, t, k) = \begin{cases} y_{\text{item},t,k}, & t \geqslant \tau_{\text{item}}, \\ g_{\text{item},t,k}, & t < \tau_{\text{item}}. \end{cases} \tag{24}$$

Algorithm 1 Update

function Update((user, item, T, l));
RecRating(user).push((user, item, T, l));
RecRating(item).push((user, item, T, l));
ConvertRating(user);
UpdateParameter(user);
ConvertRating(item);
UpdateParameter(item);

function ConvertRating(user);
while RecRating(user).size() $> M$ **do**
  (user, item$_1$, t_1, l_1) $\leftarrow$ RecRating(user).front();
  Update $\phi_{\text{user},j}$ by (23);
  RecRating(user).pop();
  (user, item$_2$, t_2, l_2) $\leftarrow$ RecRating(user).front();
  if $\tau_{\text{user}} < t_2$ **then**
    $\tau_{\text{user}} \leftarrow t_2$;
    for $j = 1, \ldots, J$ **do**
      $f_{\text{user},t,j} \leftarrow x_{\text{user},t,j}$;
    end for
  end if
end while

function UpdateParameter(user);
for $t \in$ TM(RR(user)) **do**
  Prepare $E_{\text{user},t,j}$ by (12);
end for
Initial $\alpha_{\tau_{\text{user}},j}$ as (13);
for $t \in$ TM(RR(user)), forward **do**
  Calculate $\alpha_{t,j}$ by (14);
end for
Initial $\beta_{T,j}$ as (15);
for $t \in$ TM(RR(user)), backward **do**
  Calculate $\beta_{t,i}$ by (16);
end for
for $t \in$ TM(RR(user)) **do**
  Calculate $\gamma_{t,i}$ and $\xi_{t,i,j}$ by (17) and (18);
end for
Update π_i, $A_{i,j}$, and $p_{j,k}$ by (19)–(22);

If the converted rating is the last recent rating of time t_1, there is not recent rating at t_1. The value τ_{user} changes from t_1 to the next time that user has recent ratings. In this situation, $X_{\text{user},t}$ starts at new τ_{user} and $P(X_{\text{user},t_1} = j \mid \text{Rating})$ would be an illegal value. The function $q(\text{user}, t_1, j)$ inherits its value for later use.

We describe the algorithms of computing the posterior distributions and updating the global parameters. First, we prepare $E_{\text{user},t,j}$ for $t \in \text{Time}_{\text{user}}$ according to (12) with $q(\cdot)$ replaced by $\hat{q}(\cdot)$. We compute α, β, γ and ξ according to (13)–(18) with the jump rate matrix A. Then we update the global parameters π, A and p. These parameters are computed as the divisions of sums of the values from all the users. We do not need to compute all the users' values once again. Instead, we record just the sums and the contribution of each user. When one user's value is updated, we subtract the previous contribution of the user and add the new one.

Finally, we analyze the time cost of the updating algorithm. It takes the time cost $O(JK)$ to convert to an early rating. The number of converted ratings at a time is less than M. Thus, the time cost of the conversion is $O(JKM)$. It taks the time cost $O(JKM)$ to prepare $E_{\text{user},t,j}$ for computing the posterior distributions needs. The matrix exponential and matrix logarithm have simple algorithm in the time complexity of $O(J^3)$. The posterior distributions α, β and ξ need $O(J^2M)$ and γ needs $O(JM)$. The updates of π, A and p need $O(J)$, $O(J^2M)$ and $O(JKM)$, respectively. The time cost for the tasks of items is the same as those of users, except exchanging J and K. Therefore, the total time cost for an

Table 3 The experiment datasets with different sizes

Dataset	User	Item	Rating	Density(%)
MLK (MovieLens100k) [56]	944	1683	100000	6.29
MLM (MovieLens1M)	6040	3706	1000209	4.47
Ep (Epinions) [57, 58]	2874	2624	122361	1.62
EpEx (Epinions extended) [59]	11201	109520	5449415	0.44

updating is $O(JKM + J^3M + K^3M)$. Because J, K, M are all fixed, the time cost for an updating has a constant limit. The algorithms will not increase time cost after the system running for a long time. This is an advantage of our algorithms especially with consideration of online recommendation for big data.

4.2 Prediction

The prediction works when a query $(\text{user}, \text{item}, t)$ is asked. We compute $\hat{r}_n = P(R_{\text{user,item},t} = n \mid \text{Rating})$ in Algorithm 2. If the last rating of user is at time T, after the system updated, $P(X_{\text{user},T} = j \mid \text{Rating})$ is known and stored in $x_{\text{user},T,j}$. The distributions in time $(t - T)$ are easy to find by

$$P(X_{\text{user},t} = j \mid \text{Rating}) = \sum_{i=1}^{J} P(X_{\text{user},T} = i \mid \text{Rating}) \exp(Q(t-T))_{i,j}. \tag{25}$$

In the same way, we compute $P(Y_{\text{item},t} = k \mid \text{Rating})$. The predicted rating is calculated according to (1) as

$$\begin{aligned} &P(R_{\text{user,item},t} = l \mid \text{Rating}) \\ &= \sum_{j=1}^{J} \sum_{k=1}^{K} P(X_{\text{user},t} = j \mid \text{Rating}) P(Y_{\text{item},t} = k \mid \text{Rating}) \Pr(l-1; N-1, p_{j,k}). \end{aligned} \tag{26}$$

Algorithm 2 Prediction

function Predict((user, item, t));
$(\text{user}, \text{item}_1, T_1, l_1) \leftarrow$ RecRating(user).back();
for $j = 1, \ldots, J$ **do**
  $\hat{x}_{\text{user},t,j} \leftarrow \sum_{i=1}^{J} x_{\text{user},T_1,i} \exp(A(t-T_1))_{i,j}$;
end for
$(\text{user}_2, \text{item}, T_2, l_2) \leftarrow$ RecRating(item).back();
for $k = 1, \ldots, K$ **do**
  $\hat{y}_{\text{item},t,k} \leftarrow \sum_{m=1}^{K} y_{\text{item},T_2,l} \exp(B(t-T_2))_{m,k}$;
end for
for $n = 1, \ldots, N$ **do**
  $\hat{r}_n \leftarrow \sum_{j=1}^{J} \sum_{k=1}^{K} \hat{x}_{\text{user},t,j} \hat{y}_{\text{item},t,k} \Pr(n-1; N-1, p_{j,k})$;
end for
return $\hat{r}$;

5 Experiments

We did experiment on four datasets with different sizes as illustrated in Table 3 [56–59]. The ratings of the datasets are 1–5 levels. We made clean-up preprocess to produce 20-core datasets for Ep and EpEx. The density is the number of ratings divided by the sum of numbers of users and items. Note that EpEx is a sparse dataset.

We denote RT as the algorithm of implementing our model and compare RT with the following state-of-the-art time-aware recommendation algorithms:

• Streaming recommender systems (SRec) [48]: handled data as streams for effective recommendations and used a continuous-time random process to capture dynamics of both users and items. It provided an online algorithm for real-time updating and making recommendations.

Table 4 The scores of precision

Setting	Classical				Time-ordered			
Dataset	MLK	MLM	Ep	EpEx	MLK	MLM	Ep	EpEx
SRec	0.698	0.728	0.796	0.953	0.662	0.703	0.788	0.943
TCARS	**0.717**	**0.750**	0.783	0.951	0.702	**0.724**	0.746	0.940
GRU4Rec	0.652	0.654	0.715	0.936	0.613	0.622	0.705	0.939
RRN	0.665	0.730	0.740	0.938	0.684	0.702	0.734	0.936
RT	**0.717**	0.745	**0.831**	**0.954**	**0.730**	0.717	**0.792**	**0.945**

• Time- and community-aware recommendation system (TCARS) [38]: calculated the similarity of users by both rating-based and time-weighted methods and identified the overlapping community structure among users by the similarity. It took time-weighted associative rule mining on the overlapping communities to make top-N recommendations.

• Session-based recommendations with RNN (GRU4Rec) [53]: concentrated on session-based data to provide accurate recommendations and used customized GRU model to analyze the sequence generated by a user in a session.

• Recurrent recommender networks (RRN) [51]: used an LSTM model to capture dynamics of both users and items in addition to a traditional low-rank factorization to describe the stationary components.

We make two kinds of experiments to validate the performance of the algorithms in the datasets.

• **Classical experiment.** The ratings in the datasets are randomly divided into the training set (80%) and the test set (20%).

• **Time-ordered experiment.** The ratings in the datasets are reordered according to the time they are generated. We take the former 80% as the training set and the remainder 20% as the test set. This is a reasonable setup for time-aware recommender systems for it ensures that the algorithms predict the future by the past.

The test ratings that the users or items have no ratings in the training set are not counted in the evaluation scores. In this case, the parameters of some algorithms for the users or items are not defined or initialized by small random values.

The hyperparameters of the algorithms are decided by grid search. Each hyperparameter is selected from a set of candidates to produce the best performance. For example, λ_1 and λ_2 of our model, the learning rate and regularization of RRN are selected from $\{1, 0.1, 0.01, 0.001, 0.0001\}$. The hyperparameters of comparison algorithms are selected from similar sets according to the literature. The latent vector length (for RT, SRec and RRNs) is directly related to the costs of time and space and set as 10 for every algorithm for the sake of fairness.

We compare the accuracy of the algorithms by the scores of Precision, NDCG and MRR [60]. Precision is the ratio of the positive ratings (levels 4 and 5) to the sum of positive ratings and negative ratings (levels 1–3). NDCG is normalized DCG (discounted cumulative gain), which evaluates the usefulness of an item based on its rank in a recommended list. We convert the levels 1–5 to the relevancy {0, 0.25, 0.5, 0.75, 1} for computing NDCG. MRR (mean reciprocal rank) is the average of the reciprocal ranks of the recommendation list for all users. We demonstrate the experiment results of the accuracy by the scores of Precision, NDCG and MRR in Tables 4–6, respectively. For the results, our RT has better performance on most experiments (16 of 24 scores). In other 8 scores, RT also has competitive performance in the nearly second place. TCARS has better scores for it adds the user community-aware detection beyond time-aware recommendation. In general, RT has the improvement performance if the data is time-ordered, because it gradually updates the recommendation by the distinction between the recent and early data. The results show that our model makes use of the time information in a proper way.

In addition, we experiment our RT on online learning and cold-start recommendation. In online experiment, the ratings in the datasets are reordered by the time that they are generated and imported one by one to the algorithm. For every rating, the algorithm first gives predicted rating and then updates the parameters when it receives new coming rating. The online performance is judged by RMSE (root

Table 5 Normalized discounted cumulative gain

Setting	Classical				Time-ordered			
Dataset	MLK	MLM	Ep	EpEx	MLK	MLM	Ep	EpEx
SRec	0.924	0.933	0.945	**0.984**	0.938	0.933	**0.954**	0.979
TCARS	**0.931**	**0.941**	0.936	0.979	0.939	**0.940**	0.938	0.978
GRU4Rec	0.903	0.904	0.905	0.974	0.916	0.908	0.923	0.976
RRN	0.914	0.932	0.915	0.971	0.933	0.933	0.927	0.976
RT	0.930	0.938	**0.956**	**0.984**	**0.946**	0.939	**0.954**	**0.980**

Table 6 Mean reciprocal rank

Setting	Classical				Time-ordered			
Dataset	MLK	MLM	Ep	EpEx	MLK	MLM	Ep	EpEx
SRec	0.855	0.897	0.919	**0.980**	0.850	0.872	0.897	**0.971**
TCARS	0.854	**0.910**	0.880	0.955	0.794	0.876	0.831	0.964
GRU4Rec	0.797	0.839	0.835	0.969	0.765	0.837	0.815	0.966
RRN	0.828	0.892	0.843	0.953	0.833	0.875	0.843	0.964
RT	**0.867**	0.903	**0.939**	0.978	**0.853**	**0.884**	**0.898**	0.970

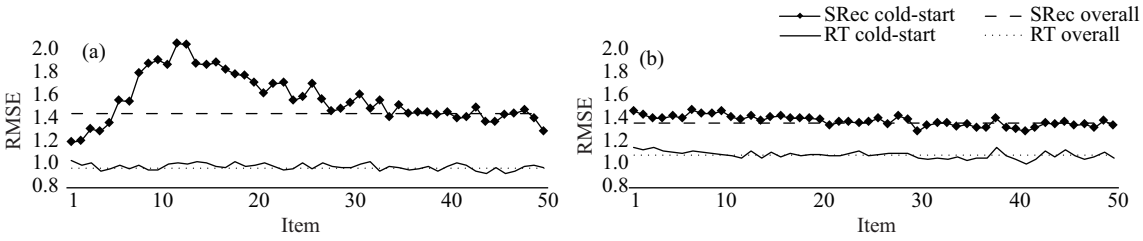

Figure 1 Online learning and cold-start performance. (a) MLK; (b) Ep.

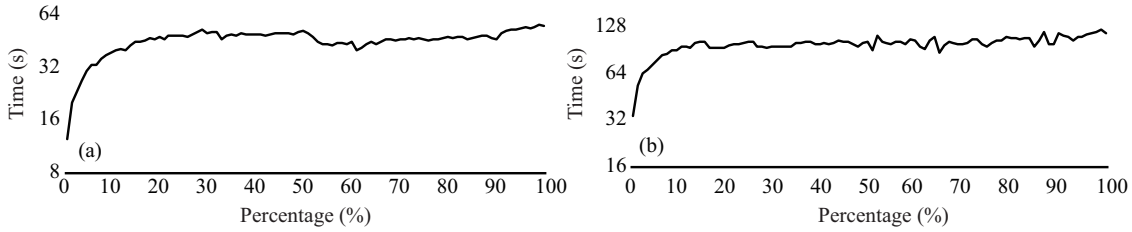

Figure 2 Time convergence. (a) MLM; (b) EpEx.

mean square error) score of the predicted ratings and true ratings. We compare RT with the online SRec to show the performance in cold-start situation. To do so, we collect the cold-start RMSE on the sequence of ratings of each user. The user cold-start RMSE is close to the overall RMSE means that the algorithm can provide new users more accurate recommendation. We illustrate the online learning and the cold-start performances of SRec and RT on MLM and Ep in Figure 1 (the coordinate axes are the items and the RMSE, respectively). The user cold-start RMSE of RT is very close to the overall RMSE than the one of SRec. Thus, RT has better cold-start performance than SRec.

Finally, we illustrate the running time curve of RT on larger datasets MLM and EpEx in Figure 2 (the coordinate axes are the percentage of data and the running time per 1% data, respectively). The running time of RT converges quickly. The algorithm does not increase time cost after running for a period of about 15% data, because the time cost for an updating has a constant limit. This makes our model specially useful for online recommendation with large scale data.

6 Conclusion

We proposed a probabilistic model that captures temporal dynamics of recommender systems. The model makes the distinction between the recent and early data in principle. For collaborative filtering, the early ratings are used to indicate the overall prior distribution of the random variable of both users and items. The recent ratings are used in the hidden Markov model to capture the change of the preferences of both the users and items over time. The model has a fast algorithm that updates every time it receives new coming data. The time cost of updating is independent to total numbers of ratings of the related users or items. The algorithm is suitable for larger scale online recommendation tasks. The experiments show that our model outperforms the existing temporal and online models for recommender systems.

Furthermore, our model can be applied to various time-aware recommender systems other than collaborative filtering. In our model, the numbers of the user types and the item types are given as hyperparameters. Because there are no suggestions about how to decide these numbers, we have to make a try to find a proper number. There would be a way to make the model adjustable for the numbers of the types of both users and items automatically. Besides, it is useful to consider the influences among users in our model. For example, friends influence interests each other. This consideration needs the variables of users not to be independent. We considered the evaluation metrics on accuracy to validate our model. The other metrics, such as diversity and novelty, are usually contrary to the accuracy. It is interesting to improve our algorithm for a trade-off among accuracy, diversity and novelty. The application of our model in a larger scale industrial online recommendation task is under investigation.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61672049, 61732001).

References

1 Koren Y. Collaborative filtering with temporal dynamics. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009. 447–456
2 Baltrunas L, Amatriain X. Towards time-dependant recommendation based on implicit feedback. In: Proceedings of Workshop on Context-aware Recommender Systems, New York, 2009. 423–424
3 Xiang L, Yang Q. Time-dependent models in collaborative filtering based recommender system. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, Milan, 2009. 450–457
4 Agarwal D, Chen B C, Elango P. Fast online learning through offline initialization for time-sensitive recommendation. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010. 703–712
5 Koenigstein N, Dror G, Koren Y. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In: Proceedings of the 5th ACM Conference on Recommender Systems, 2011. 165–172
6 Ramirez-Garcia X, Garcia-Valdez M. A pre-filtering based context-aware recommender system using fuzzy rules. In: Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization. Berlin: Springer, 2015. 497–505
7 Suksawatchon J, Darapisut S, Suksawatchon U. The constant time of predictive algorithm for music recommendation with time context. In: Proceedings of International Joint Conference on Computer Science and Software Engineering, 2015. 63–68
8 Bogina V, Kuflik T, Mokryn O. Learning item temporal dynamics for predicting buying sessions. In: Proceedings of the 21st Annual Meeting of the Intelligent Interfaces, Sonoma, 2016. 251–255
9 Xiong L, Chen X, Huang T K, et al. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In: Proceedings of the SIAM International Conference on Data Mining, 2010. 211–222
10 Dunlavy D M, Kolda T G, Acar E. Temporal link prediction using matrix and tensor factorizations. ACM Trans Knowl Discov Data, 2011, 5: 1–27
11 Bhargava P, Phan T, Zhou J Y, et al. Who, what, when, and where: multi-dimensional collaborative recommendations using tensor factorization on sparse user-generated data. In: Proceedings of International World Wide Web Conferences Steering Committee, 2015. 130–140
12 Géry M, Haddad H. Evaluation of web usage mining approaches for user's next request prediction. In: Proceedings of the 5th ACM International Workshop on Web Information and Data Management, New Orleans, 2003. 74–81
13 Huang Y M, Huang T C, Wang K T, et al. A Markov-based recommendation model for exploring the transfer of learning on the web. J Educ Tech Soc, 2009, 12: 144–162
14 Awad M A, Khalil I. Prediction of user's web-browsing behavior: application of Markov model. IEEE Trans Syst Man Cybern B, 2012, 42: 1131–1142
15 Hariri N, Mobasher B, Burke R. Context-aware music recommendation based on latenttopic sequential patterns. In: Proceedings of ACM Conference on Recommender Systems, 2012. 131–138

16 Chen W, Niu Z D, Zhao X Y, et al. A hybrid recommendation algorithm adapted in e-learning environments. World Wide Web, 2014, 17: 271–284
17 Zhang J D, Chow C Y. Point-of-interest recommendations in location-based social networks. SIGSPATIAL Special, 2016, 7: 26–33
18 Chen J, Wang C K, Wang J M. A personalized interest-forgetting Markov model for recommendations. In: Proceedings of the 29th AAAI Conference on Artificial Intelligence, 2015. 16–22
19 Gopalachari M V, Sammulal P. Hybrid recommender system with conceptualization and temporal preferences. In: Proceedings of the 2nd International Conference on Computer and Communication Technologies, 2015. 811–819
20 Sahoo N, Singh P V, Mukhopadhyay T. A hidden Markov model for collaborative filtering. Mis Quart, 2012, 36: 1329–1356
21 Sanchez F, Alduan M, Alvarez F, et al. Recommender system for sport videos based on user audiovisual consumption. IEEE Trans Multimedia, 2012, 14: 1546–1557
22 Alanazi A, Bain M. A people-to-people content-based reciprocal recommender using hidden markov models. In: Proceedings of the 7th ACM Conference on Recommender Systems, 2013. 303–306
23 Gu W R, Dong S B, Zeng Z Z. Increasing recommended effectiveness with markov chains and purchase intervals. Neural Comput Applic, 2014, 25: 1153–1162
24 Zhang H D, Ni W C, Li X, et al. Modeling the heterogeneous duration of user interest in time-dependent recommendation: a hidden semi-Markov approach. IEEE Trans Syst Man Cybern Syst, 2018, 48: 177–194
25 Zhang H, Ni W, Li X, et al. A hidden semi-Markov approach for time-dependent recommendation. In: Proceedings of Pacific Asia Conference on Information Systems, 2016
26 Le D T, Fang Y, Lauw H W. Modeling sequential preferences with dynamic user and context factors. In: Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Riva del Garda, 2016. 145–161
27 Alanazi A, Bain M. A scalable people-to-people hybrid reciprocal recommender using hidden Markov models. In: Proceedings of the 2nd International Workshop on Machine Learning Methods for Recommender Systems, 2016
28 Lu Z, Agarwal D, Dhillon I S. A spatio-temporal approach to collaborative filtering. In: Proceedings of the 3rd ACM Conference on Recommender Systems, New York, 2009. 13–20
29 Paisley J, Gerrish S, Blei D. Dynamic modeling with the collaborative Kalman filter. In: Proceedings of the 5th Annual NYAS Machine Learning Symposium, 2010
30 Sun J Z, Varshney K R, Subbian K. Dynamic matrix factorization: a state space approach. In: Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing, 2012. 1897–1900
31 Sun J Z, Parthasarathy D, Varshney K R. Collaborative Kalman filtering for dynamic matrix factorization. IEEE Trans Signal Process, 2014, 62: 3499–3509
32 Gultekin S, Paisley J. A collaborative Kalman filter for time-evolving dyadic processes. In: Proceedings of IEEE International Conference on Data Mining, Shenzhen, 2014. 140–149
33 Ding Y, Li X. Time weight collaborative filtering. In: Proceedings of the ACM International Conference on Information and Knowledge Management, 2005. 485–492
34 Liu N N, Zhao M, Xiang E, et al. Online evolutionary collaborative filtering. In: Proceedings of the 4th ACM Conference on Recommender Systems, Barcelona, 2010. 95–102
35 Chandramouli B, Levandoski J J, Eldawy A, et al. StreamRec: a real-time recommender system. In: Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data, 2011. 6–8
36 Huang Y X, Cui B, Zhang W Y, et al. TencentRec: real-time stream recommendation in practice. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, 2015. 227–238
37 Su H Y, Lin X F, Yan B, et al. The collaborative filtering algorithm with time weight based on mapReduce. In: Proceedings of International Conference on Big Data Computing and Communications, 2015. 386–395
38 Rezaeimehr F, Moradi P, Ahmadian S, et al. TCARS: time- and community-aware recommendation system. Future Generation Comput Syst, 2018, 78: 419–429
39 Yu H, Li Z Y. A collaborative filtering method based on the forgetting curve. In: Proceedings of the 2010 International Conference on Web Information Systems and Mining, Sanya, 2010. 183–187
40 Shi Y C. An improved collaborative filtering recommendation method based on timestamp. In: Proceedings of International Conference on Advanced Communication Technology, 2014. 1180–1184
41 Xiang L, Yuan Q, Zhao S W, et al. Temporal recommendation on graphs via long- and short-term preference fusion. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2010. 723–731
42 Ding Y, Wang D, Li G Q, et al. Exploiting long-term and short-term preferences and RFID trajectories in shop recommendation. Softw Pract Exper, 2017, 47: 849–865
43 Yu J, Liu F F. A short-term user interest model for personalized recommendation. In: Proceedings of IEEE International Conference on Information Management and Engineering, Chengdu, 2010. 219–222
44 Li L, Zheng L, Yang F, et al. Modeling and broadening temporal user interest in personalized news recommendation. Expert Syst Appl, 2014, 41: 3168–3177
45 Basile P, Caputo A, Gemmis M D, et al. Modeling short-term preferences in time-aware recommender systems. In: Proceedings of Workshop on Deep Content Analytics Techniques for Personalized and Intelligent Services, 2015. 44–54
46 Daneshmand S M, Javari A, Abtahi S E, et al. A time-aware recommender system based on dependency network of

items. Comput J, 2015, 58: 1955–1966
47 Azadjalal M M, Moradi P, Abdollahpouri A, et al. A trust-aware recommendation method based on Pareto dominance and confidence concepts. Knowledge-Based Syst, 2017, 116: 130–143
48 Chang S Y, Zhang Y, Tang J L, et al. Streaming recommender systems. In: Proceedings of the 26th International Conference on World Wide Web, Perth, 2017. 381–389
49 Tan Y K, Xu X X, Liu Y. Improved recurrent neural networks for session-based recommendations. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 2016. 17–22
50 Devooght R, Bersini H. Long and short-term recommendations with recurrent neural networks. In: Proceedings of Conference on User Modeling, Adaptation and Personalization, 2017. 13–21
51 Wu C Y, Ahmed A, Beutel A, et al. Recurrent recommender networks. In: Proceedings of the 10th ACM International Conference on Web Search and Data Mining, Cambridge, 2017. 495–503
52 Villatel K, Smirnova E, Mary J, et al. Recurrent neural networks for long and short-term sequential recommendation. 2018. ArXiv: 1807.09142
53 Hidasi B, Karatzoglou A. Recurrent neural networks with top-k gains for session-based recommendations. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, 2018. 843–852
54 Durrett R. Essentials of Stochastic Processes. New York: Springer, 2012. 139–183
55 Rabiner L R. A tutorial on hidden Markov models and selected applications in speech recognition. Proc IEEE, 1989, 77: 257–286
56 Harper F M, Konstan J A. The MovieLens datasets. ACM Trans Interact Intell Syst, 2016, 5: 1–19
57 Tang J L, Gao H J, Liu H. Mtrust: discerning multi-faceted trust in a connected world. In: Proceedings of the 5th ACM International Conference on Web Search and Data Mining, 2012. 93–102
58 Tang J L, Liu H, Gao H J, et al. Etrust: understanding trust evolution in an online world. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012. 253–261
59 Massa P, Avesani P. Trust-aware recommender systems. In: Proceedings of the 2007 ACM Conference on Recommender Systems, New York, 2007. 17–24
60 Jalili M, Ahmadian S, Izadi M, et al. Evaluating collaborative filtering recommender algorithms: a survey. IEEE Access, 2018, 6: 74003–74024