A Contextual Collaborative Approach for App Usage Forecasting

Yingzi Wang[*†1], **Nicholas Jing Yuan**[§2], **Yu Sun**[‡†3], **Fuzheng Zhang**[†4], **Xing Xie**[†5], **Qi Liu**[*6], **Enhong Chen**[*7]
[*]University of Science and Technology of China [†]Microsoft Research
[§]Microsoft Corporation [‡]University of Melbourne
[1]yingzi@mail.ustc.edu.cn, {[2]nicholas.yuan, [4]fuzzhang, [5]xing.xie}@microsoft.com,
[3]sun.y@unimelb.edu.au, {[6]qiliuql, [7]cheneh}@ustc.edu.cn

ABSTRACT
Fine-grained long-term forecasting enables many emerging recommendation applications such as forecasting the usage amounts of various apps to guide future investments, and forecasting users' seasonal demands for a certain commodity to find potential repeat buyers. For these applications, there often exists certain homogeneity in terms of similar users and items (e.g., apps), which also correlates with various contexts like users' spatial movements and physical environments. Most existing works only focus on predicting the upcoming situation such as the next used app or next online purchase, without considering the long-term temporal co-evolution of items and contexts and the homogeneity among all dimensions. In this paper, we propose a contextual collaborative forecasting (CCF) model to address the above issues. The model integrates contextual collaborative filtering with time series analysis, and simultaneously captures various components of temporal patterns, including trend, seasonality, and stationarity. The approach models the temporal homogeneity of similar users, items, and contexts. We evaluate the model on a large real-world app usage dataset, which validates that CCF outperforms state-of-the-art methods in terms of both accuracy and efficiency for long-term app usage forecasting.

Author Keywords
app usage forecasting; tensor decomposition; seasonal time series; collabotative filtering

INTRODUCTION
Long-term forecasting is a typical way to predict future values for many traditional applications. For example, it is used to predict the trends of stock prices to minimize risks and maximize returns for investors [4], to predict the demand change of certain commodity for manufacturers [12], and to predict severe weather such as heavy rain and blizzards to reduce economic losses [2].

UbiComp '16, September 12-16, 2016, Heidelberg, Germany
© 2016 ACM. ISBN 978-1-4503-4461-6/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2971648.2971729

Many newly emerged applications further require fine-grained long-term forecasting. Specifically, to provide more accurate and personalized recommendation, we need to understand users' evolving preferences in terms of similar users, similar items, and various contexts. For example, online retailers need to know similar users' long-term seasonal demand for certain commodity and hence find potential repeat buyers [32]; App developers need to find users' long-term preferences for different apps to guide future investments; And news providers need to understand users' time-varying preferences for topics under various contexts. There are also many other ubiquitous applications that require fine-grained long-term forecasting considering similarities and contexts. For example, when forecasting the count of influenza-like-illness cases for different countries [37], we could explore similarities among regions and diseases, and find results under various contexts (e.g., time-varying regional demographics and economic statuses). When forecasting the long-term probability of failures for thousands of servers in a data center, we could consider similar performances of servers and various contexts measured by temperature, humidity, etc. From the above examples, we can observe that different from traditional forecasting problems, these applications present evident homogeneous patterns, for example, users with similar society background or lifestyles tend to aggregate into groups [15, 39], as with items with similar product functions [19, 20] or similar information contents [22]. In addition, these applications are also related to various contexts, such as users' spatial movements and seasonal weather changes, which reflect real-life situations and enable more accurate forecasting.

Most previous works have only attempted to predict users' upcoming requirement at current contexts (location, time, previous demand, etc.). For example, some methods predict the next app a user will use to improve device usability and optimize hardware operation [1, 28, 39], or predict customers' next online purchase to prepare for next item recommendation [17, 34]. These methods lack the ability to model long-term pattens when only predicting the upcoming requirement one by one because the uncertainty of future contexts will bring significant challenges when we apply these methods to long-term forecasting.

Timeseries analysis methods, for example, the ARIMA algorithm [3] and Discrete Fourier Transform [18], present inspiring performance in long-term forecasting for a single time series. However, these methods cannot effectively address fine-grained long-term forecasting problem due to the following reasons. First, real-world user behavior data are often sparse and long-tail. Second, these methods lack the capability to exploit users' contextual information. Thirdly, they can not model the behavior patterns of users without history data, which is considered a data missing problem. Although a few studies have incorporated the collaborative filtering approach in their time series models (tackling the sparse and missing data problems), such as MLDS in [26] and TriMine in [23], they ignore the seasonal and trending properties, and contextual information in temporal patterns.

In this paper, we propose a scalable and generalized **C**ontextual **C**ollaborative **F**orecasting (**CCF**) model to detect long-term patterns and jointly forecast the future values of many variables of interest. Specifically, we deploy a high-dimensional collaborative filtering method through tensor decomposition to tackle sparse and data missing problems, where we exploit the aggregate properties of similar users, items, and contexts. We apply a detrend seasonal auto-regressive approach to predict the variables for each user in each type of context, considering the auto-regressive, seasonal, and trending properties simultaneously for a long period. We systematically integrate collaborative filtering and time series analysis through a joint optimization approach, accomplishing the temporal co-evolution of all informative dimensions (i.e., user, item, and context). In particular, we investigate and develop the CCF model with the specific application of long-term app usage forecasting. The proposed model could be easily applied to many other long-term forecasting problems such as modeling users' long-term online shopping behavior.

In summary, our paper offers the following contributions.

- We propose the CCF model which systematically integrates contextual collaborative method with time series analysis. The temporal co-evolution of informative dimensions is achieved by an innovative joint optimization.
- We simultaneously consider seasonal, trending, and autoregressive properties of long-term temporal data, and model the homogeneity of all informative dimensions using a collaborative filtering approach.
- We explore the correlation between users' app usage behaviors and various types of contexts, and observe that app usage patterns in functional-analogous venues are more similar than venues of different functions.
- We evaluate the CCF model using a large real-world dataset which consists of top-100 dominantly used apps (covering over 70% share of the app market) and users' real-time locations. Experiment results demonstrate that CCF significantly outperforms state-of-the-art algorithms for forecasting and data missing problems on several metrics.

The rest of paper is organized as follows. We first review related work, and then introduce the contextual and homogeneous patterns, define the long-term forecasting problem, and present the CCF model. After that, we report experiments and conclude our work.

RELATED WORK

App Usage Prediction and Recommendation

We currently have abundant apps that provide useful services in almost every aspect of modern life. Both users and app developers have increasing requirements to explore the usage patterns of different apps and users, for individual and commercial use, respectively. Several studies have demonstrated that human mobility is highly predictable in both spatial and temporal dimensions [6, 36], especially for mobile phone usage patterns [29]. Therefore, app usage prediction [10, 28, 39] becomes a meaningful and achievable application for user patterns mining on mobile apps.

Spatial and temporal mobility patterns are two fundamental aspects in predicting future activities. Huang et al. [10] provide a Bayesian network and a linear model to explore the spatialtemporal influence on app usage prediction. A classification method is applied in [1] where real-time location-time contexts are considered important features. Liao et al. [18] investigate the relation between apps and their usage times. They utilize Discrete Fourier Transform to analyze usage periods and specific times of different apps. Moreover, many studies emphasize the correlation between temporally sequential app usage records, in which a Markov chain [1], a Gaussian based method [10], or a naive Bayes classifier [28] is applied to extract the relationships between app actions.

Sociality is another fundamental factor of app usage patterns [15]. Xu et al. [39] demonstrate that people in the same community tend to share similar patterns of making app usage decisions, especially for those users sharing similar spatial-temporal lifestyles. The collaborative filtering technique is an effective method to detect similar users and explore individual potential app usage tendentiousness [13, 31, 40], especially in a sparse dataset. Nevertheless, most previous studies focus on the next app usage prediction, ignoring the long-term trend of apps for different users. Similar to users, apps for similar situations or services also present similar long-term spatial-temporal usage patterns.

Tensor Decomposition and Time Evolution

The tensor decomposition method has strong performance in personalized prediction and recommendation applications [23, 25, 30, 42]. By computing a low rank approximation of the original tensor, tensor decomposition clusters related dimensions into specified numbers of groups. Zheng et al. [42] map user, location, and user activity into a 3-dimensional tensor to address the sparse problem in recommendation systems. Rendle et al. [25] introduce a gradient descent optimization method to solve the ranking problem for user tag recommendation instead of the traditional least-square method. However, real-world data are seldom stationary, and traditional tensor decomposition algorithms lack the ability to deal with temporal dynamic problems.

Time evolution investigation is a profound aspect in long-term forecasting for weather, economics, supply chain, user mobility prediction, and so on [2, 4, 12, 27]. Some fundamental and effective time series analyzing methods, like ARIMA, Discrete Fourier Transform, and Markov chain algorithms, are prevalently adopted in temporal dynamic analysis [3, 10,

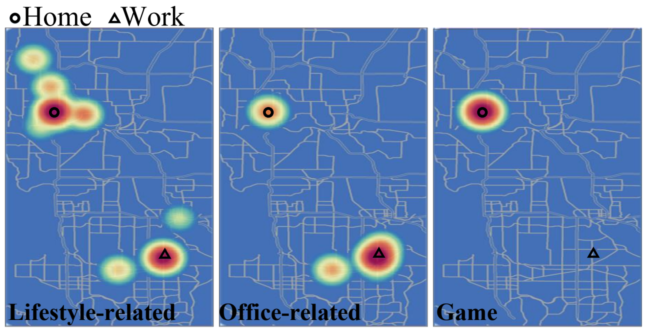


Figure 1. Spatial distributions of three different app usage logs

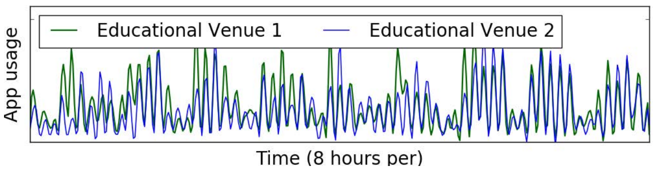
18]. Tseng et al. [33] propose a hybrid forecasting model, combining the seasonal ARIMA method and the neural back propagation model. Elfeky et al. [7] and Vlachos et al. [35] explore the algorithms to detect periodicity in time-series data, capturing the evolution patterns of temporal data.

After a long-time development of tensor decomposition and time evolution algorithms, researchers perceive the limitation of considering only one of them in some time-varying dataset. A Bayesian probabilistic tensor decomposition method [38] makes a Markovian assumption for the time-dependent feature vector, assuming that each time vector depends only on its immediate predecessor. Similarly, Rogers et al. [26] present a multilinear dynamic system to model time series patterns in a tensor, considering that the data in a time series are multilinear projections of the latent matrices in tensor decomposition. By combining auto-regressive model and tensor decomposition, Matsubara et al. [23] propose a forecasting method, TriMine, for time-evolving datasets. Unfortunately, they only deal with Markovian correlated and auto-regressive properties in time series, ignoring other important patterns, like seasonality and tendency. Besides, TriMine firstly decomposes the useritemtime tensor into three latent matrices and then exploits the auto-regressive model on temporal matrix for multiple time window sizes, having the risk of losing temporal patterns in the initial tensor decomposition procedure.

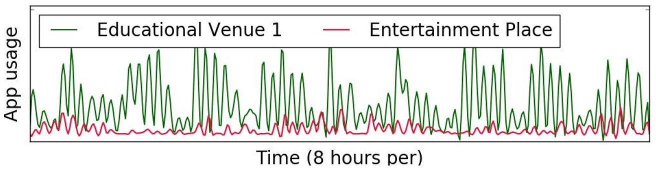
PROBLEM FORMULATION

Contextual and Homogeneous Patterns of App Usage

Many studies have demonstrated that human mobility is highly predictable in a spatial-temporal space [6, 36]. Apps on mobile devices, which are "carried" with people most time and are easily triggered, can effectively reflect users' mobility characteristics [29]. Users exhibit diverse moving patterns for different apps in spatial context. Figure 1 shows an anonymous user's spatial distribution when he/she launches three different apps (a lifestyle-related app, an office-related app, and a game app) in a city of USA, from November 20, 2015 to January 23, 2016. Here the black *circles* represent the user's home, and the *triangles* are his/her work place. We can clearly see that lifestyle-related apps primarily distribute around the locations of home and office, and scatter around some other places close to home and office. Work place and home are dominant places for office-related apps and game apps, respectively. When users are in different contexts, the purpose in that situation probably leads to a certain suitable app. For instance, users will launch a food recommendation app when he/she steps into a food court, but when they enter a hospital, they may open the hospital's on-line registration system soon.



(a) The app usage in two functionally similar venues



(b) The app usage in two functionally different venues

Figure 2. App usage patterns of a study-related app at three different venues contexts

The homogeneous effects of similar users and apps have been investigated in recent studies [15, 39]. Users with similar society backgrounds and preferences attempt to aggregate into a group since they may have analogous mobility patterns. Besides, a person will typically become the user of a number of apps with overlapping service purposes, to build broader within-app social relationships, or enjoy diverse service experiences. For example, a user may use Facebook and Twitter frequently during the same period, which are both social apps, and a white-collar worker who launches Microsoft Word everyday may have high demand for Excel. For apps, the ones with similar functions, even similar user interface styles, may attract the users with similar lifestyles or requirements. Also, it is well understood that the usage of apps of similar functionalities shows similar usage patterns. Work-related apps have more users during working hours while game apps attract more users in evenings and weekends. In addition, we discover that aggregate patterns also exist in the contextual dimension. Figure 2 presents the temporal usage amounts of a study-related app in two education-related venues (green and blue lines) and one entertainment place (red line). It illustrates that the app usage in two functional-analogous venues (education-related) is more similar than two venues of different functions (an education-related venue and an entertainment venue). Both venue visits and app usages are purpose-driven activities, which helps us understand the similar homogeneous pattens of visit and usage behaviors.

Problem Definition

We define our studied problem in this section. First, we introduce the notations used throughout this paper. Let $\mathcal{U} = \{u_1, u_2, ..., u_M\}$ denote the M users. Let $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ represent the N apps in our model. Let $C = \{c_1, c_2, ..., c_L\}$ represent the L spatial-contextual features of a certain user-app-time group (detailed in Data Preprocessing Section). We split the time line into a number of *slices*, for example, 8 hours per slice. Let time slice list $\mathcal{S} = \{s_1, s_2, ..., s_{T_S}\}$ denote the training period and $\mathcal{F} = \{f_1, f_2, ..., f_{T_F}\}$ denote the testing period, where f_1 is the next slice of s_{T_S}.

Our studied problem is to forecast the long-term app usage amount and trend for each user-app pair, where the long-term

Table 1. Mathematical Notation

Symbol	Size	Description
$\boldsymbol{\mathcal{X}}$	$M \times N \times L \times T_S$	detrend training tensor of app usage for M users, N apps, L contexts and T_S time slices
$\widetilde{\boldsymbol{\mathcal{X}}}$	$M \times N \times L \times T_F$	forecasting tensor
$\ddot{\boldsymbol{\mathcal{X}}}$	$M \times N \times L \times T_F$	testing tensor
$\mathbf{U}$	$M \times K$	user latent matrix
$\mathbf{V}$	$N \times K$	app latent matrix
$\mathbf{C}$	$L \times K$	context latent matrix
$\mathbf{S}$	$T_S \times K$	detrend training time series latent matrix
$\mathbf{F}$	$T_F \times K$	forecasting time series latent matrix
$\mathbf{W}^{(k)}$	$T \times H^{(k)}$	constraint regularization matrix for $\mathbf{S}_k$
$\mathbf{\Phi}$	$K \times P$	time series' AR parameter matrix
$\mathbf{\Theta}$	$K \times Q$	time series' seasonal parameter matrix
$\boldsymbol{d}$	$1 \times K$	season length set for K clusters

means at least half a month. Specifically, given the app usage amount of all dimensions u_m, v_n, c_l and s_{t_S} (where $u_m \in \mathcal{U}$, $v_n \in \mathcal{V}$, $c_l \in C$ and $s_{t_S} \in \mathcal{S}$), our goal is to extract the patterns to forecast the temporal fluctuation of a certain user u_m and app v_n in future time slices $\mathcal{F}$. For model simplicity, we assume that slices in $\mathcal{S}$ and $\mathcal{F}$ have the same split length. Moreover, we can forecast the temporal trend in various time granularities (4 hours or 6 hours per time slice etc) to capture diverse time-related characteristics.

MODEL

We propose a constrained tensor decomposition model to systematically integrate the contextual collaborative filtering technique with time series analysis. Specifically, we utilize a high-dimensional tensor to exploit aggregate properties of similar users, apps and contexts. Then, we apply a detrend seasonal auto-regressive approach to constrain and model the temporal dimension of this tensor, considering the auto-regressive, seasonal, and trending properties simultaneously with a joint optimization.

4-Dimensional Tensor Decomposition

To capture the complicated temporal relationships among users, apps, and contexts, as well as the homogeneous phenomenon in each dimension as mentioned before, we introduce a 4-dimensional tensor $\boldsymbol{\mathcal{X}}$ in our model. Each entry $\boldsymbol{\mathcal{X}}_{m,n,l,t}$ represents the app usage amount of app n for user m in the l^{th} context at time slice t. We assume that each dimension contains K latent clusters. Specifically, we decompose tensor $\boldsymbol{\mathcal{X}}$ into 4 low-rank latent matrices $\mathbf{U}$, $\mathbf{V}$, $\mathbf{C}$ and $\mathbf{S}$, which is illustrated in Figure 4. For a better visulization, we flatten the time dimension to a parallel direction with context in Figure 4. Column vectors $\mathbf{U}_k$, $\mathbf{V}_k$, $\mathbf{C}_k$ and $\mathbf{S}_k$ represent the k^{th} cluster of each latent matrix. Noted that the co-evolving temporal patterns, such as regularity or sequentiality of time series, constrain the distribution of $\mathbf{S}$ matrix. We deploy a temporal regularization function $\mathcal{G}(\mathbf{S})$ to achieve such pattern-related constraints, where $\mathcal{G}(\mathbf{S}) = 0$ if the time series latent matrix $\mathbf{S}$ satisfies the constraint completely. The larger $\mathcal{G}(\mathbf{S})$ is, the less $\mathbf{S}$ meets the constraint. More details about $\mathcal{G}(\mathbf{S})$ are discussed in the following section. Now we use CP decomposition [16] to decompose tensor $\boldsymbol{\mathcal{X}}$ into rank-one tensors and estimate $\boldsymbol{\mathcal{X}}$ as:

$$\boldsymbol{\mathcal{X}} \approx \sum_{k=1}^{K} \mathbf{U}_k \circ \mathbf{C}_k \circ \mathbf{V}_k \circ \mathbf{S}_k, \quad s.t. \;\; \mathcal{G}(\mathbf{S}) = 0, \tag{1}$$

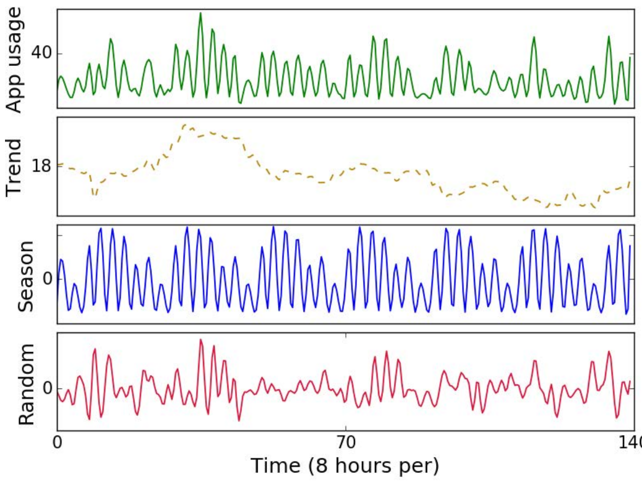


Figure 3. Decomposition of an app usage time series

where $(\mathbf{U}_k \circ \mathbf{C}_k \circ \mathbf{V}_k \circ \mathbf{S}_k)_{u,c,v,s} = \mathbf{U}_{u,k}\mathbf{C}_{c,k}\mathbf{V}_{v,k}\mathbf{S}_{s,k}$, and $\circ$ represents the vector outer product [16]. We summarize the related notations and their sizes in Table 1.

Seasonal AR Parameters

A single time series includes all of or part of the three components: trend, seasonality and stationarity [3, 9]. Trend is the slow, gradual increasing or decreasing patterns over the whole time series. Seasonality, also called periodicity, is the recurring component with regular moving characteristic, which shows regular fluctuation in a time series curve. Stationarity is defined by the condition that $(X_1, ..., X_n)$ and $(X_{1+h}, ..., X_{n+h})$ have the same joint distribution for any integer h and $n > 0$ [3]. Figure 3 is the decomposition result of a certain user's app usage time series using moving averages method in [14]. The time series is decomposed into three components as mentioned above. The top figure is the raw app usage count series in 46 days (from Nov. 20 2015), where a time slice contains 8 hours. The yellow, blue, and red lines (from the second to the last figure) represent the trend, seasonal, and stationary components of the raw time series, respectively. We can clearly see that the seasonal component shows a one-week (21 time slices) periodical pattern, with higher app usage in weekdays and lower in weekends. The trend comes up in the first two weeks then gradually decreases in the rest time.

Time series analytics techniques often first remove the trend and seasonality from the raw time series, estimate the remaining stationary component [3], compute the trend and period separately, and then add them back to the stationary component. Following this method, we apply a *trend identifying* method [8] and a *detrending* approach [3] to remove the trends from raw time series. Specifically, we apply linear regression on all time series against time and compute the slope coefficient for estimating significance. If the coefficient is larger than a predefined threshold, we consider that the time series includes a linear trend component. Next, we eliminate all the trend components in raw time series by differencing. In particular, for each time series $\bar{\boldsymbol{\mathcal{X}}}_{m,n,l}$ with a trend component, the difference operation of each time slice t is defined as:

$$\boldsymbol{\mathcal{X}}_{m,n,l,t} = \nabla \bar{\boldsymbol{\mathcal{X}}}_{m,n,l,t} = \bar{\boldsymbol{\mathcal{X}}}_{m,n,l,t} - \bar{\boldsymbol{\mathcal{X}}}_{m,n,l,t-1}, \tag{2}$$

for $t > 1$.

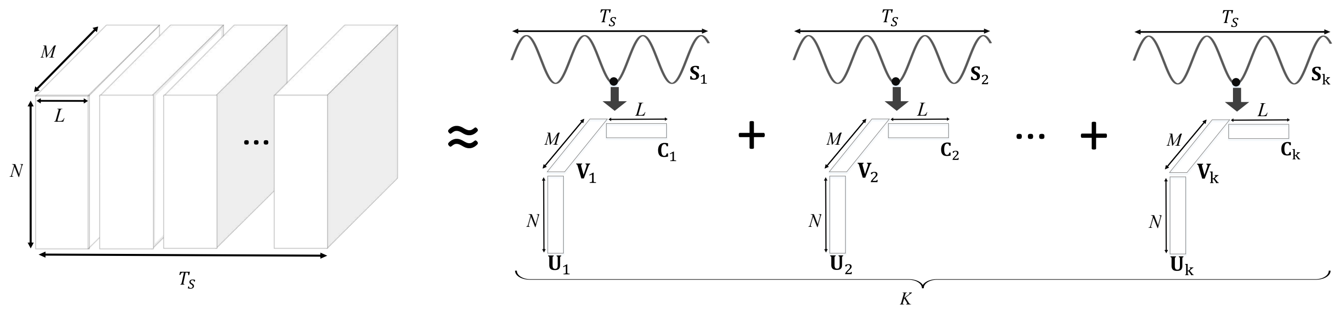


Figure 4. Illustration of CCF model

Nevertheless, the seasons of different user-app-context groups are diverse and complex, which makes it difficult to detect them, even remove them one by one. As illustrated before, the temporal co-evolving series in tensor $\boldsymbol{\mathcal{X}}$ tend to aggregate into K clusters, forming the latent matrix $\mathbf{S}$. We propose to investigate the seasonal and stationary components of the K temporal clusters through a seasonal auto-regressive (AR) algorithm [3, 5, 9, 33], and obtain the seasonal and autoregressive components simultaneously. If we only consider the AR part of the model, the constraint function $\mathcal{G}(\mathbf{S})$ of time series $\mathbf{S}_k$ can be written as:

$$\phi_k(B) \cdot \mathbf{S}_{k,t} = Z_t^k, \tag{3}$$

where

$$\begin{cases} \phi_k(B) = 1 - \Phi_{k,1}B - \Phi_{k,2}B^2 - ... - \Phi_{k,P}B^P, \\ B^p\mathbf{S}_{k,t} = \mathbf{S}_{k,t-p}, \\ Z_t^k \sim WN(0, \sigma_k^2), \\ t \in \{P+1, P+2, ..., T_S\}. \end{cases}$$

Here we assume $T_S > (P+1)$. P is the auto-regressive degree in the model, which indicates that the current result is correlated with the previous P time slices $(\mathbf{S}_{k,t-1}, \mathbf{S}_{k,t-2}, ..., \mathbf{S}_{k,t-P})$. $\Phi_{k,p}$ is the correlation coefficient for time slice $\mathbf{S}_{k,t-p}$, representing the weight of each previous time slice, and $\phi_k(\cdot)$ is a P^{th} polynomial. B is the backshift operater and Z_t^k is white noise.

Now we discuss the seasonal component of the previous function. We apply the Segment Periodicity Detection (SPD) method [7] to discover the periodic pattern of a time series without completely searching the whole sequence. Specifically, we first reduce the dimensionality of time series and discretize them using the method in [21], transforming the values of time slices into a predefined number of symbols, and then apply SPD on the K latent time series and obtain the period set $\boldsymbol{d} = \{d_1, d_2, ..., d_K\}$. We add the seasonal difference and seasonal AR component to the constraint function $\mathcal{G}(\mathbf{S})$:

$$\phi_k(B) \cdot \theta_k(B^{d_k}) \cdot (1 - B^{d_k})^D \mathbf{S}_{k,t} = Z_t^k, \tag{4}$$

where

$$\begin{cases} \theta_k(B^{d_k}) = 1 - \Theta_{k,1}B^{d_k} - \Theta_{k,2}B^{2d_k} - ... - \Theta_{k,Q}B^{Qd_k}, \\ t \in \{(Q+D)d_k + P + 1, (Q+D)d_k + P + 2, ..., T_S\}, \end{cases}$$

and other parameters conform the definitions in Equation (3). Here we also assume $T_S > ((Q+D)d_k+P+1)$. Q is the seasonal auto-regressive degree, which indicates that the current value is relevant to previous Q seasons, and D is a non-negative integer represents the seasonal difference degree (similar with the difference operation in Equation (2)). Brockwell and Davis [3] suggest that D is rarely more than 1, and P, Q are less than 3 in typical applications. Therefore, we set $D = 1$ and P, Q less than or equal to 3 in the rest of our paper.

Let $\Phi_{k,0} = -1, \Theta_{k,-1} = 0, \Theta_{k,0} = -1$ and $\Theta_{k,Q+1} = 0$, Equation (4) can be rewritten as

$$\sum_{q=0}^{Q+1} \sum_{p=0}^{P} (\Theta_{k,q} - \Theta_{k,q-1})\Phi_{k,p}B^{qd_k+p}\mathbf{S}_{k,t} = Z_t^k. \tag{5}$$

CCF Model

With the above formulation, we can write the function $\mathcal{G}(\mathbf{S})$ in Equation (1) as:

$$\mathcal{G}(\mathbf{S}) = \sum_{k=1}^{K} \sum_{t=\bar{t}}^{T_S} \sum_{q=0}^{Q+1} \sum_{p=0}^{P} (\Theta_{k,q} - \Theta_{k,q-1})\Phi_{k,p}B^{qd_k+p}\mathbf{S}_{k,t} - Z_t^k, \tag{6}$$

where $\bar{t} = (Q+1)d_k + P + 1$. $\mathcal{G}(\mathbf{S}_k) = 0$ represents the seasonal auto-regressive constraint function for latent time series $\mathbf{S}_k$. At this point, we have specified the constraint function $\mathcal{G}(\mathbf{S})$ of latent matrix $\mathbf{S}$ to achieve obtaining the seasonal and autoregressive components simultaneously.

To integrate the estimation of tensor $\boldsymbol{\mathcal{X}}$ and the constraint function $\mathcal{G}(\mathbf{S})$ for temporal co-evolution, we propose a constraint matrix $\mathbf{W}^{(k)} \in R^{T_S \times (T_S-(Q+1)d_k-P)}$ for $\mathbf{S}_k$. $\mathcal{G}(\mathbf{S}) = 0$ is equivalent to $\sum_{k=1}^{K} \|\mathbf{S}_k^\top \mathbf{W}^{(k)}\|_F^2 = 0$. Assuming $P, Q < d_k$, we can define $\mathbf{W}^{(k)}$ as:

$$\mathbf{W}_{i,j}^{(k)} = \begin{cases} \Psi(q,p) & if \quad (i-j) \bmod d_k = P - p \\ & and \quad (i-j)/d_k = Q + 1 - q\,, \\ 0 & otherwise \end{cases} \tag{7}$$

here $\Psi(q,p) = (\Theta_{k,q} - \Theta_{k,q-1})\Phi_{k,p}$. The matrix structure of $\mathbf{W}^{(k)}$ is:

$$\begin{pmatrix} \Psi(Q+1, P\sim 0) & & & \\ \vdots & \ddots & & \\ \Psi(1, P\sim 0) & & \Psi(Q+1, P\sim 0) & \\ \vdots & \ddots & \vdots & \Psi(Q+1, P\sim 0) \\ \Psi(0, P\sim 0) & & \Psi(1, P\sim 0) & \vdots \\ & \ddots & \vdots & \Psi(1, P\sim 0) \\ & & \Psi(0, P\sim 0) & \vdots \\ & & & \Psi(0, P\sim 0) \end{pmatrix},$$

where $\Psi(q, P\sim 0) = (\Psi(q,P), \Psi(q,P-1), ..., \Psi(q,0))^\top$, and the intervel between $\Psi(q, P\sim 0)$ and $\Psi(q+1, P\sim 0)$ is $(d_k - P)$.

Algorithm 1: Optimization of CCF Model

```
Input: $\bar{\mathcal{X}}, K, \lambda, \eta, I, iter, \epsilon$
Output: $\mathbf{U}, \mathbf{V}, \mathbf{C}, \boldsymbol{d}, \boldsymbol{\Phi}, \boldsymbol{\Theta}$
1  detect $\bar{\mathcal{X}}_{m,n,l}$ with trend, update it with (2);
2  $\mathbf{U}, \mathbf{V}, \mathbf{C}, \mathbf{S}, \boldsymbol{d} \Leftarrow \mathbf{U}_0, \mathbf{V}_0, \mathbf{C}_0, \mathbf{S}_0, \boldsymbol{d}_0$ ;
3  i = 0;
4  repeat
5      i++;
6      update U with (9);
7      update V with (11);
8      update C with (10);
9      for $k = 1, 2, ..., K$ do
10         update $\mathbf{S}_k$ with (12);
11     update $\boldsymbol{d}$;
12     estimate $\boldsymbol{\Phi}, \boldsymbol{\Theta}$ using maximum likelihood procedure ;
13     for $k = 1, 2, ..., K$ do
14         update $\mathbf{W}^{(k)}$ with (7);
15 until $(\Omega$ in $(4) < \epsilon)$ or $(i > iter)$;
16 return $U, V, C, d, \boldsymbol{\Phi}, \boldsymbol{\Theta}$
```

If we consider $\Psi(q, P \sim 0)$ as a module, each column of $\mathbf{W}^{(k)}$ contains $(Q + 2)$ discontinuous modules.

Now our goal is to estimate parameters $\mathbf{U}$, $\mathbf{V}$, $\mathbf{C}$, $\mathbf{S}$, and $\{\mathbf{W}^{(1)}, \mathbf{W}^{(2)}, ..., \mathbf{W}^{(K)}\}$ that minimize the following objective function:

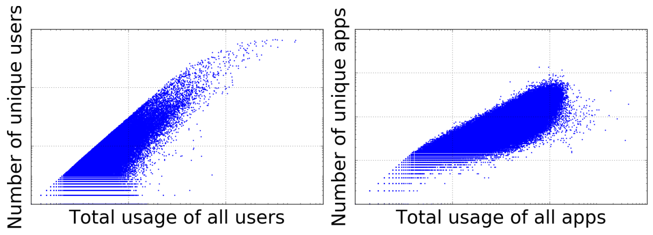
$$\Omega = \|\mathcal{X} - \sum_{k=1}^{K} \mathbf{U}_k \circ \mathbf{C}_k \circ \mathbf{V}_k \circ \mathbf{S}_k\|_F^2 + \lambda \sum_{k=1}^{K} \|\mathbf{S}_k^\top \mathbf{W}^{(k)}\|_F^2 + \eta(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2 + \|\mathbf{C}\|_F^2), \tag{8}$$

where $(\mathbf{U}_k \circ \mathbf{C}_k \circ \mathbf{V}_k \circ \mathbf{S}_k)_{u,c,v,s} = \mathbf{U}_{u,k}\mathbf{C}_{c,k}\mathbf{V}_{v,k}\mathbf{S}_{s,k}$, and λ, η are two regularization parameters. The optimization of matrix $\mathbf{W}^{(k)}$ is the estimation of $\boldsymbol{\Phi}$, $\boldsymbol{\Theta}$, and $\boldsymbol{d}$, which are the parameters of time series' seasonal and AR components.

After obtaining all the parameters above, we can effectively forecast the future app usage amount. We estimate the future time series latent matrix $\mathbf{F} \in R^{T_F \times K}$using Equation (5), and compute the forecasting tensor as $\widetilde{\mathcal{X}} = \sum_{k=1}^{K} \mathbf{U}_k \circ \mathbf{C}_k \circ \mathbf{V}_k \circ \mathbf{F}_k$.

Optimization

Tensor decomposition can be approximated as a linear least-square problem [16]. However, the optimization of our latent matrices is more complicated on account of the regularization terms. Therefore, we apply the alternative estimation method in our model optimization. We iteratively estimate the four tensor latent matrices $\mathbf{U}$, $\mathbf{V}$, $\mathbf{S}$, $\mathbf{C}$, and the constraint matrix $\mathbf{W}^{(k)}$, systematically integrating collaborative filtering approach and time series analysis method. We use gradient descent to estimate the four latent matrices. It is hard to compute the gradient of a 4-dimensional tensor directly. Thus we firstly flatten the tensor $\mathcal{X}$ into matrix along different dimensions, and then compute the gradient of each matrix. Specifically, we flatten the tensor through 4 dimensions which are denoted by $\mathcal{X}_{(U)}, \mathcal{X}_{(C)}$, $\mathcal{X}_{(V)}$, and $\mathcal{X}_{(S)}$, respectively. Specifically, matrix $\mathcal{X}_{(U)} \in R^{M \times (TNL)}$ is a type of arrangement of $\mathcal{X}$, where the user



(a) distribution of users (b) distribution of apps

Figure 5. Statistics of app usage data

dimension in $\mathcal{X}$ becomes the rows of the resulting matrix $\mathcal{X}$ and the rest dimensions are flattened into a vector through the oreder of $\{context, app, time\}$. Other matrices are obtained similarly.

Let $\hat{\mathcal{X}}$ denotes the vector outer product of four latent matrices $\sum_{k=1}^{K} \mathbf{U}_k \circ \mathbf{V}_k \circ \mathbf{C}_k \circ \mathbf{S}_k$. We flatten $\hat{\mathcal{X}}$ and obtain the following matrices:

$$\hat{\mathcal{X}}_{(U)} = \mathbf{U}(\mathbf{S} \odot \mathbf{V} \odot \mathbf{C})^\top \in R^{M \times (TNL)};$$
$$\hat{\mathcal{X}}_{(C)} = \mathbf{C}(\mathbf{S} \odot \mathbf{V} \odot \mathbf{U})^\top \in R^{L \times (TNM)};$$
$$\hat{\mathcal{X}}_{(V)} = \mathbf{V}(\mathbf{S} \odot \mathbf{C} \odot \mathbf{U})^\top \in R^{N \times (TLM)};$$
$$\hat{\mathcal{X}}_{(S)} = \mathbf{S}(\mathbf{V} \odot \mathbf{C} \odot \mathbf{U})^\top \in R^{T \times (NLM)},$$

where $\odot$ is the *Khatri-Rao product* [16]. We compute the gradients of the above four target matrices:

- Gradient of $\mathbf{U}$:

$$\frac{\partial \Omega}{\partial \mathbf{U}} = 2(\mathcal{X}_{(U)} - \hat{\mathcal{X}}_{(U)})(-\mathbf{S} \odot \mathbf{V} \odot \mathbf{C}) + 2\eta \mathbf{U}, \tag{9}$$

- Gradient of $\mathbf{C}$:

$$\frac{\partial \Omega}{\partial \mathbf{C}} = 2(\mathcal{X}_{(C)} - \hat{\mathcal{X}}_{(C)})(-\mathbf{S} \odot \mathbf{V} \odot \mathbf{U}) + 2\eta \mathbf{C}, \tag{10}$$

- Gradient of $\mathbf{V}$:

$$\frac{\partial \Omega}{\partial \mathbf{V}} = 2(\mathcal{X}_{(V)} - \hat{\mathcal{X}}_{(V)})(-\mathbf{S} \odot \mathbf{C} \odot \mathbf{U}) + 2\eta \mathbf{V}, \tag{11}$$

- Gradient of $\mathbf{S}_k$:

$$\frac{\partial \Omega}{\partial \mathbf{S}_k} = 2(\mathcal{X}_{(S)} - \hat{\mathcal{X}}_{(S)})(-\mathbf{V}_k \odot \mathbf{C}_k \odot \mathbf{U}_k) + 2\lambda \mathbf{W}^{(k)}\mathbf{W}^{(k)\top}\mathbf{S}_k, \tag{12}$$

where $\mathbf{W}^{(k)}$ is defined in Equation (7).

We need to further estimate $\boldsymbol{\Phi}$, $\boldsymbol{\Theta}$ and $\boldsymbol{d}$ for $\mathbf{W}^{(k)}$. The approximation of $\boldsymbol{d}$ is presented in Detrend Seasonal Auto-Regression Section. The $\boldsymbol{\Phi}$ and $\boldsymbol{\Theta}$ vectors are estimated by the maximum likelihood procedure [3]. Pseudo-code of the optimization process is presented in Algorithm 1.

EXPERIMENT

Settings

Data Preprocessing

We utilized a real-world app usage dataset from a digital assistant consisting of anonymized users' PC and mobile phone records from November 20, 2015 to January 23, 2016. The information of each record included user id, app name, app category, launch time, duration time, time zone, and etc. Using only PC records or mobile phone records fails to reflect users' total demands on apps, so we use PC and mobile data simultaneously. Users could link their PCs and mobile phones through uniform accounts. In addition, considering that only mobile phones could reflect users' real-time locations, we selected users with at least one mobile phone record. We mainly collected users' app usage logs and location signal logs. We plot the Log-Log distribution of these users and used apps in Figure 5. Figure 5(a) shows that a small amount of apps have dominant usage while a large amount of apps were launched less than 1000 times. Figure 5(b) shows that most users have similar app usage patterns during the 65 days, and only a few of them have extremely high or low app usage records.

We also collected POI venues' information (latitude, longitude, and category) of United States and a few countries in the Middle East. We then mapped locations signals of users in these areas to POI venues' location information to obtain the visited venue signals. According to our statistics, the top-100 apps have dominant usage records (more than 70%) in these areas. To experiment with apps having sufficient records, we chose the app usage records of the these top-100 apps in our experiments. After that, the detection method in [41] was applied to detect the home and work places for these users. Then, we filtered users with less than 3250 usage logs of these 100 apps (50 records per day on average), and less than 65 visited venue logs (visited some venues at least once a day on average). We also filtered users who only used less than 50 apps in the 100 apps, avoiding getting an excessively sparse dataset. After such preprocessing, we obtained 11,489,997 app usage logs of 1558 users, together with their app usage location records, visited venues, home, and work information.

Baselines

We compare the proposed CCF model with four baselines, MLDS, TriMine, Seasonal ARIMA, and Context-aware next app prediction model.

- MLDS. Multilinear Dynamical System (MLDS) [26] models the time series tensor as a multilinear projection on some latent spaces. Specifically, it creates a latent tensor sequence $\{\mathcal{Z}_1, \mathcal{Z}_2, ..., \mathcal{Z}_{T_S}\}$, in which each tensor $\mathcal{Z}_t$ is projected to an observation tensor $\boldsymbol{\mathcal{X}}_t$. The initialization probability $P(\mathcal{Z}_1)$, the conditional distribution probability $P(\mathcal{Z}_{t+1}|\mathcal{Z}_t)$ and the observation probability $P(\boldsymbol{\mathcal{X}}_t|\mathcal{Z}_t)$ are estimated through an EM algorithm.
- TriMine. TriMine [23] assumes each latent matrix of tensor $\boldsymbol{\mathcal{X}}$ has K hidden topics. It uses topic modeling approach and applies the collapsed Gibbs sampling method to extract the latent factors for each dimension. Then they model the temporal dimension matrix with auto-regression method on multiple time granularities.
- SARIMA. Seasonal ARIMA (SARIMA) is an efficient non-stationary single time series analysis algorithm [3, 11]. It forecasts each single time series of a certain user-app pair through the automatic SARIMA approach proposed in [11].
- Next-Pre. Contextual next app prediction (Next-Pre) method collects several user-related, environment-related, and app-related contextual data, estimating the probability of each app through a naive Bayes classifier [28]. In our experiments, we use Next-Pre to predict the next app one by one keeping consistent with the app usage frequency in training data.

Metrics

We use three metrics to measure the performance of app usage forecasting: Root Mean Square Error, Relative Euclidean Distance, and Pearson' Correlation. Since we only focus on user-app long-term forecasting, we define the user-app testing data and forecasting result as $\ddot{\mathbf{X}}$, $\widetilde{\mathbf{X}} \in R^{M\times N\times T_F}$, respectively. For CCF model, we define $\widetilde{\mathbf{X}}_{m,n,t} = \sum_{l=1}^{L} \widetilde{\boldsymbol{\mathcal{X}}}_{m,n,l,t}$ and $\ddot{\mathbf{X}}_{m,n,t} = \sum_{l=1}^{L} \ddot{\boldsymbol{\mathcal{X}}}_{m,n,l,t}$

- RMSE. Root Mean Square Error (RMSE) [24] of time slice t is defined as:

$$RMSE_{(t)} = \sqrt{\sum_{(m,n)} (\ddot{\mathbf{X}}_{m,n,t} - \widetilde{\mathbf{X}}_{m,n,t})^2 / |\widetilde{\mathbf{X}}_t|},$$

  where $t \in \{1, 2, ..., T_F\}$, and $|\widetilde{\mathbf{X}}_t|$ is number of elements in the t^{th} time slice of $\widetilde{\mathbf{X}}$.

- RED. Relative Euclidean Distance (RED) [26] is another metric to measure the app usage count prediction error:

$$RED_{(t)} = \sqrt{(\sum_{(m,n)} (\ddot{\mathbf{X}}_{m,n,t} - \widetilde{\mathbf{X}}_{m,n,t})^2) / (\sum_{m,n,l} \ddot{\mathbf{X}}^2_{m,n,t})}.$$

- PC. We exploited the Pearson' Correlation (PC) to measure the trend similarity (the similarity of curves' shapes):

$$PC_{(t)} = \frac{E[\ddot{\mathbb{X}}_t \tilde{\mathbb{X}}_t] - E[\ddot{\mathbb{X}}_t]E[\tilde{\mathbb{X}}_t]}{\sqrt{E[\ddot{\mathbb{X}}_t^2] - E[\ddot{\mathbb{X}}_t]^2}\sqrt{E[\tilde{\mathbb{X}}_t^2] - E[\tilde{\mathbb{X}}_t]^2}}$$

  where $t \in \{I, I+1, ..., T_F\}$ and $\ddot{\mathbb{X}}_t$ is a I-length sequence defined as $\ddot{\mathbb{X}}_{t,i} = \sum_{m,n} \ddot{\mathbf{X}}_{m,n,t-I+i}$. In our experiments, we set $I = 10$;

Here, a lower RMSE, RED value and a higher PC value indicate a better performance.

Experiment Design

We used the location information and visited venues as app usage context. The locations where a user launched an app were categorized into four types: home-correlated, office-correlated, venue-correlated and others. We allocated app usage locations to the first or second type if their distances to home or office were less than a chosen threshold (500 meters in our experiment). Subsequently, we aggregated visited venues into 18 general types according to the POI information, such as travel, education, restaurant, and art, and mapped all the venue-visited-synchronous app signals into these general types. The app signals which did not match to home, work or any venues were categorized into the "others" type. Therefore, the context dimension contained 21 spatial and venue-related types of content. We split the 65 days between November 20, 2015 and January 23, 2016 into 195 time slices, with 8 hours per slice. We use 140 slices as *training* data and 55 as *testing*.

RMSE	CCF	CCF-Con	CCF-appCF	CCF-Ss	CCF-CoEv	TriMine	SARIMA	Next-Pre
12-8am	**0.8362**	0.8651	0.8472	1.0766	0.8725	1.0649	1.5053	2.8716
8am-4pm	**2.1390**	2.2125	2.2233	2.2033	2.2108	2.5369	2.7291	4.1295
4pm-12am	**1.8802**	1.9370	1.9752	1.9092	1.9498	2.2069	2.4505	4.0407

Table 2. RMSE results of different models.

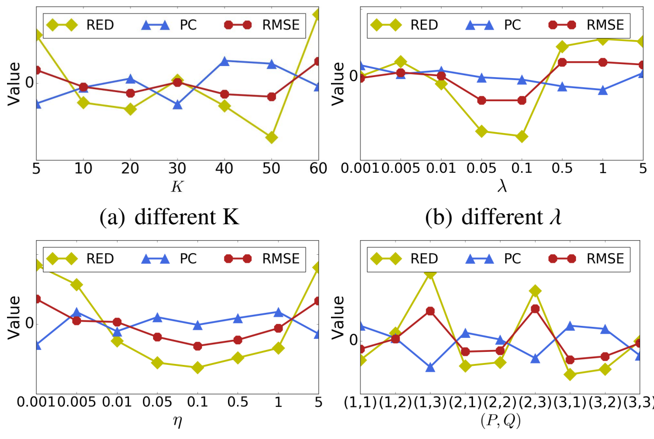


Figure 6. Experiment results of CCF for different parameters

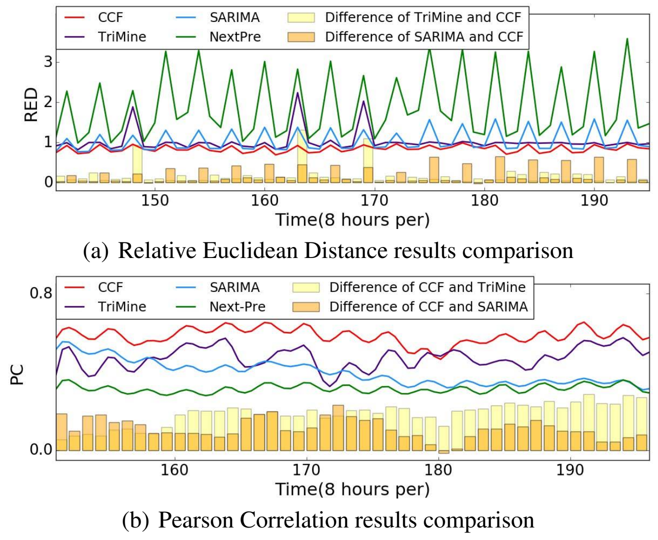


Figure 7. Experiment results of CCF, SARIMA, TriMine, and Next-Pre

In this way, we created a training tensor $\bar{\mathcal{X}} \in R^{1558\times21\times100\times140}$ and a testing tensor $\ddot{\mathcal{X}} \in R^{1558\times21\times100\times55}$.

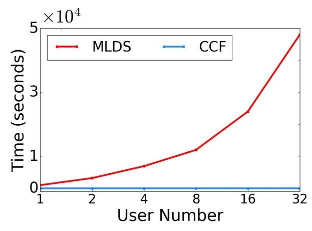
To emphasize the effects of different components in CCF, we add four sub-CCF models in the experiment: 1. We remove the seasonal part of CCF and denote it by **CCF-Ss**, 2. We change the co-evolving optimization procedure in Algorithm 1, only updating matrix $\mathbf{W}^{(k)}$ after the $iter^{th}$ optimization iterations, which is represented as **CCF-CoEv**, 3. We aggregate the contextual dimension of tensor $\bar{\mathcal{X}}$ and $\ddot{\mathcal{X}}$, denote it as **CCF-Con**, 4. We ignore the collaborative filtering of different apps and optimize the result for each app separately, which is denoted as **CCF-appCF**.

Forecasting the app usage trend for a user without history records is required in many real-world applications. Therefore, we also designed a data missing test in our experiment, which randomly removed a certain proportion of user-app pairs in training data, and evaluated the performance of CCF and TriMine against these removed data (we cannot evaluate SARIMA and Next-Pre since they can only forecast the app usage of a user with complete history records).

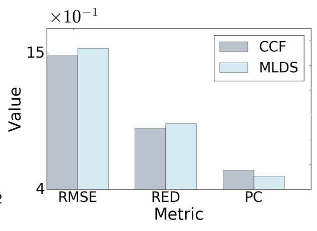
Results

We first present the results of different parameter values in Figure 6. We applied z-normalization approach to amplify and normalize the results. Figure 6(a) plots the result when we vary K from 5 to 60 when $(\lambda, \eta) = (0.1, 0.1)$ and $(P, Q) = (3, 1)$. We can see that the CCF model has the best performance when $K = 50$, and the sudden decrease of performance happens when K is larger than 50. Factor λ is the regularization parameter of time series latent matrix $\mathbf{S}$. Figure 6(b) shows that the influence of λ is stable when it is less than 0.01 and larger than 0.5, and CCF has the lowest RED and RMSE values when $\lambda = 0.1$. We can also see that the PC result shows tiny changes when λ varies from 0.001 to 5, indicating that the weight of $\mathbf{S}$'s regularization has little impact on shape similarity of forecasting series than forecasting value accuracy. Figures 6(c) and 6(d) plot the result of varying η from 0.001 to 5 and result of different (P, Q) combinations, respectively. The CCF model has the best performance on RED and PC when η (the regularization term of $\mathbf{U}$, $\mathbf{V}$ and $\mathbf{C}$) equals 0.1. The PC value keeps stable when η is between 0.05 and 1. We evaluate all the (P, Q) combinations with P and Q less than 4 (reason is explained in the Model Section). CCF shows the best results when $P = 3$ and $Q = 1$, which means that current app usage amount is highly related to the previous three time slices and one season, where three time slices are just one day. We train the CCF model with $K = 10, \lambda = 0.1, \eta = 0.1$ and $(P, Q) = (3, 1)$ in the rest of our experiment (we set K as 10 due to the model complexity problem, the complexity of CCF is linear with K).

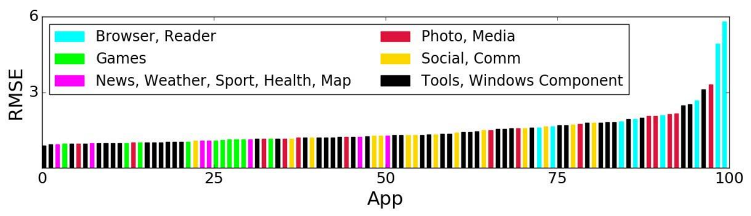
Next, we present the results of CCF, TriMine, MLDS, SARIMA, and Next-Pre. The time complexity of MLDS is extremely high for a 4-dimensional tensor. We therefore selected small sub-tensors of $\mathcal{X}$ including different amount of users and tested the efficiencies of MLDS and CCF. Figure 8(a) shows that with the growth of users, the running time of MLDS increases exponentially, while the time of CCF only presents linear growth. MLDS will take weeks of time if the number of users is larger than 1000. Therefore, we randomly selected 100 users to compare the performance of MLDS and CCF, and presented the average results of 5 runs in Figure 8(b). CCF outperforms MLDS on all the three metrics, and has a large (9.92%) improvement on PC. Figures 7(a) and 7(b) show the RED and PC results for app usage forecasting of TriMine, SARIMA, Next-Pre, and CCF, where the differences between CCF, TriMine, and SARIMA are presented in the bottom histograms. We present the RMSE results in Table 2. Overall, we observe that the performance of Next-Pre is worse than the



(a) Wall clock time comparison between CCF and MLDS

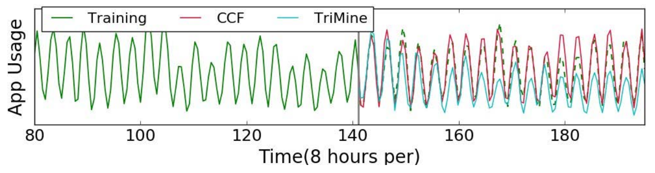


(b) Performance comparison between CCF and MLDS

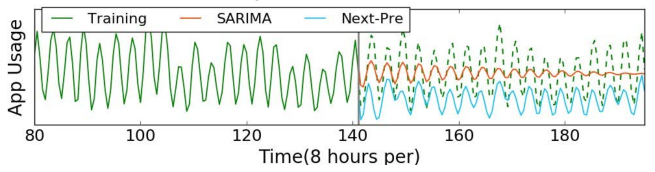


(c) RMSE values comparison among the top-100 apps

Figure 8. (a)Efficiency comparison between CCF and MLDS (b) Performance comparison between CCF and MLDS (c) RMSE values of the 100 apps

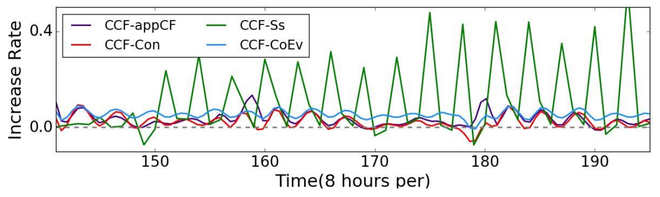


(a) Forecasting results of CCF and TriMine

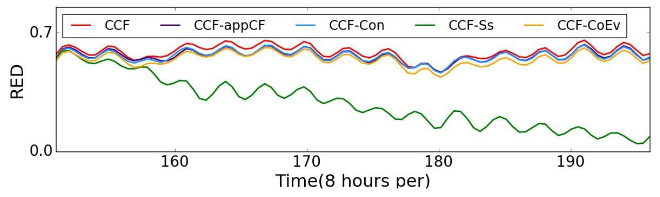


(b) Forecasting results of SARIMA and Next-Pre

Figure 9. Forecasting results of a certain life-related app from CCF, SARIMA, TriMine, and Next-Pre



(a) RED Increase rates of different sub-CCFs compared with CCF



(b) Pearson Correlation results of different sub-CCF models

Figure 10. Comparison between CCF and sub-CCF models

other three methods on RED, PC, and RMSE. Next-Pre also has an upward tend on RED, which is mainly due to the error accumulation when predicting the next app one by one. On average, CCF outperforms TriMine, SARIMA and Next-Pre by 14.75%, 15.30%, 48.63% on RED, and 14.80%, 29.36%, 57.47% on RMSE, respectively. Figure 7(b) illustrates that the forecasting PC of SARIMA deteriorates rapidly during the 55 time slices, and finally has almost the same value as Next-Pre. We can also observe that all four models have the highest RED value in the first 8 hours during a day, but lowest RMSE during that period. This demonstrates that users' app usages are smallest during 12am-8am, which is consistent with our everyday experiences. Similarly, the lowest RED and highest RMSE values during 8am-4pm verify that 8am-4pm has the largest usage amount during a day. CCF has the most stable RED and highest PC values within the testing period, which indicates the best ability to fit the trend of temporal patterns. Figure 9 shows the forecasting results of different models for a music app. For a better visualization, we separate Figure 9 into two sub-figures. The performance of TriMine is as good as CCF in the first 15 time slices, but decreases rapidly in the rest. The forecasting from SARIMA has regular and diminishing fluctuation, consistent with the increasing RED and decreasing PC values in Figure 7.

Figure 10 and Table 2 show the results of CCF and sub-CCF models. Figure 10(a) presents the increase rate of RED when we remove different components from CCF model. It clearly shows that CCF has significant superiority compared to all the sub-models. CCF-Ss has the highest RED increase rate (increase 11.27% compared with CCF on average), highest RMSE value in 12am-8am, and lowest, decreasing PC values, which confirms the importance of considering seasonal part of time series in forecasting. The removal of the co-evolving procedure during optimization, CCF-CoEv, decreases 6.96% PC from CCF, demonstrating the effectiveness of modeling co-evolution among different dimensions. CCF outperforms CCF-appCF by 3.25% on RMSE on average. The decrease of RMSE during 8am-4pm is 1.93 times larger than that of 12am-8am period, which is mainly due to the higher app usage amount which provides more collaborative opportunities.

The collaborative component of CCF model iteratively aggregates apps with similar usage patterns into clusters, detects the temporal patterns of these clusters, and then forecasts their future patterns after convergence. Figure 11 shows app usage time series of two different app clusters, where solid lines represent training data, and dotted lines are forecasting results. We can observe what the first cluster, which is plotted by the green line, contains apps that are mostly office related tools, such as Calendar and Microsoft Office. We can see a clear one-week season and a decreasing usage on weekends and American holidays, like Thanksgiving, Christmas and New Year. Besides, these apps have evident usage peak at 8am-4pm during workdays, which is consistent of our empirical thoughts of work time during a day. As mentioned before, $P = 3$ and $Q = 1$ makes the forecasting only related to the previous three time slices and one season, which leads to the lower prediction value for Thursday and Friday (due to the low app usage amount at New Year). The red line shows a cluster consisting of popular game apps, such as League of Legends and Candy Crush. Different from the first cluster, the game cluster has high usage amount on weekends and holidays. The period 4pm-12am is the usage peak of this cluster, which indicates that most users prefer to playing games during leisure time.

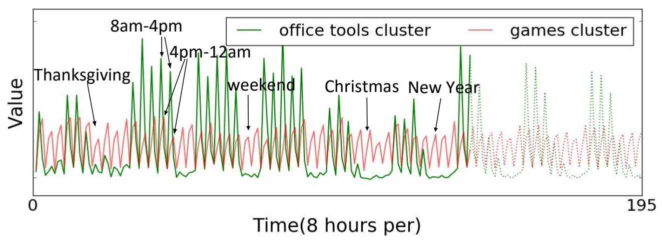


Figure 11. Time series of two app clusters

Rate	CCF			TriMine		
	RMSE	RED	PC	RMSE	RED	PC
0.1	1.7836	0.9464	0.5584	1.8391	1.0180	0.3661
0.2	1.9588	0.9494	0.5616	2.0198	1.0195	0.3717
0.3	1.9502	0.9618	0.5266	2.0091	1.0137	0.3559
0.4	1.9951	0.9553	0.5252	2.0585	1.0097	0.3453
0.5	1.9983	0.9533	0.5195	2.0632	1.0137	0.2349
0.6	2.0064	0.9627	0.4799	2.0718	1.0199	0.2222

Table 3. Data missing test results of CCF and TriMine.

We plot the average RMSE value of the 100 selected apps in Figure 8(c). We categorize these apps into six types, annotated in the figure. It shows that games and lifestyle related apps (news, weather, sport etc.) are mostly predictable, while the usages of browsers and readers are least predictable. The predictability of apps for photo, media, and tools do not have a consistent pattern, which requires more exact category division to investigate their properties. Social and communication apps, such as Facebook and Skype, show medium predictive performance, which indicates that online social and chatting behaviors has both high regularity and uncertainty.

Finally, we discuss the data missing problem of app usage forecasting, with the results presented in Table 3. The missing rate is the proportion of removed training data, and the evaluation is only tested on removed user-app pairs. We can observe that both CCF and TriMine perform worse on new users for existing apps compared with the results in Figure 7 and Table 2, and they all have decreasing performance when the missing rate increases. Nevertheless, CCF still outperforms TriMine at all missing rates on three metrics, and has a 78.43% improvement on PC compared with TriMine.

DISCUSSION

Implications and Limitations

We have evaluated our approaches on a large-scale real-world dataset, collected the usage logs of top 100 used apps which cover over 70% share of market in the digital assistant in United States and several Middle Eastern countries, and recorded the continuous locational context information of all users. Although there are some biases focusing on only top-100 apps, it still has the power of persuasion. Besides, the CCF model has superior scalability, because we can easily enlarge context types by extending the contextual vector or adding new dimensions in the tensor. Moreover, CCF can also generalize to other analogous long-term mobility modeling problems. For example, users' online shopping behaviors present various time series patterns for different products, where aggregate effects may present in either users with similar interests or products of similar functions, and long-term Point of Interests (POI) visiting mobility may also show analogous temporal properties among similar users and POIs.

Meanwhile, our model and experiment have several limitations. First, there are some biases in our dataset. We have only collected the data of users who had mobile logs and provided permission to record their locations which are only a small part of all users, ensuring the contextual dimension but losing the comprehensiveness of data. We also have only focused on forecasting top-100 apps, leaving the evaluation suffering from bias. In addition, we have considered the POI venues' information only from the United States and a few countries in the Middle East, constraining the diversity of users and apps. Some regionally famous apps, such as WeChat of China, were not included in our experiment. Second, only a small part of app usage records present conspicuous increasing or decreasing trends within two months. In the future, we will extend the time span of dataset and study more complicated interactions among different apps, such as the positive and negative correlation of different apps' usage amount tendency. Third, all the aggregated time series are modeled by a uniform latent factor size K, auto-regression degree P, seasonal difference D, and auto-regression degree Q. Our future work is to extend CCF to investigate the automatic degree learning approach, and achieve the co-evolution among different apps in optimization.

Privacy

In our experiment, we have collected the app usage logs and location signals of users who granted recording permission to the digital assistant. All the logs were anonymously recorded, and a character id with uniform format and consistent length was allocated to each user. Particularly, the system uses several advanced techniques to protect users' privacies, and deletes dataset periodically to only preserve the latest data for improving personal services of the digital assistant.

CONCLUSION

In this paper, we introduced the influence of location-based contextual information on long-term app usage patterns, and developed a general model integrating collaborative method and time series analysis, which achieves the temporal co-evolution in model optimization. In particular, we have applied collaborative filtering to exploit the homogeneous patterns of similar users, items, and contexts, and explored the long-term temporal patterns of these three dimensions through a time series analysis method considering the seasonal, trending, and auto-regressive properties simultaneously. Extensive evaluations were provided to validate the performance of our model using a large-scale real-world app usage dataset. The results have shown that our model significantly outperforms state-of-the-art methods on long-term forecasting and data missing problems.

ACKNOWLEDGMENTS

This research was partially supported by grants from the National Science Foundation for Distinguished Young Scholars of China (Grant No. 61325010). Qi Liu gratefully acknowledges the support of the Youth Innovation Promotion Association of CAS (No. 2014299) and the MOE-Microsoft Key Laboratory of USTC.

REFERENCES

1. Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 285–294.
2. Veronica J Berrocal, Adrian E Raftery, Tilmann Gneiting, and Richard C Steed. 2012. Probabilistic weather forecasting for winter road maintenance. *J. Amer. Statist. Assoc.* (2012).
3. Peter J Brockwell and Richard A Davis. 2006. *Introduction to time series and forecasting*. Springer Science & Business Media.
4. QiSen Cai, Defu Zhang, Bo Wu, and Stehpen CH Leung. 2013. A novel stock forecasting model based on fuzzy time series and genetic algorithm. *Procedia Computer Science* 18 (2013), 1155–1162.
5. Ching-Fu Chen, Yu-Hern Chang, and Yu-Wei Chang. 2009. Seasonal ARIMA forecasting of inbound air travel arrivals to Taiwan. *Transportmetrica* 5, 2 (2009), 125–140.
6. Yves-Alexandre de Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. 2013. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports* 3 (2013).
7. Mohamed G Elfeky, Walid G Aref, and Ahmed K Elmagarmid. 2005. Periodicity detection in time series databases. *Knowledge and Data Engineering, IEEE Transactions on* 17, 7 (2005), 875–887.
8. Charles Thomas Haan. 2002. Statistical methods in hydrology. (2002).
9. James Douglas Hamilton. 1994. *Time series analysis*. Vol. 2. Princeton university press Princeton.
10. Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. 2012. Predicting mobile application usage using contextual information. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 1059–1065.
11. Rob J Hyndman, Yeasmin Khandakar, and others. 2007. *Automatic time series for forecasting: the forecast package for R*. Technical Report. Monash University, Department of Econometrics and Business Statistics.
12. Sanjita Jaipuria and SS Mahapatra. 2014. An improved demand forecasting method to reduce bullwhip effect in supply chains. *Expert Systems with Applications* 41, 5 (2014), 2395–2408.
13. Alexandros Karatzoglou, Linas Baltrunas, Karen Church, and Matthias Böhmer. 2012. Climbing the app wall: enabling mobile app discovery through context-aware recommendations. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2527–2530.
14. Maurice George Kendall and others. 1946. The advanced theory of statistics. *The advanced theory of statistics.* 2nd Ed (1946).
15. Isabel Kloumann, Lada Adamic, Jon Kleinberg, and Shaomei Wu. 2015. The lifecycles of apps in a social ecosystem. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 581–591.
16. Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.
17. Farshad Kooti, Kristina Lerman, Luca Maria Aiello, Mihajlo Grbovic, Nemanja Djuric, and Vladan Radosavljevic. 2016. Portrait of an Online Shopper: Understanding and Predicting Consumer Behavior. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. ACM, 205–214.
18. Zhung-Xun Liao, Po-Ruey Lei, Tsu-Jou Shen, Shou-Chung Li, and Wen-Chih Peng. 2012. Mining temporal profiles of mobile applications for usage prediction. In *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*. IEEE, 890–893.
19. Jovian Lin, Kazunari Sugiyama, Min-Yen Kan, and Tat-Seng Chua. 2013. Addressing cold-start in app recommendation: latent user models constructed from twitter followers. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 283–292.
20. Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.
21. JLEKS Lonardi and Pranav Patel. 2002. Finding motifs in time series. In *Proc. of the 2nd Workshop on Temporal Data Mining*. 53–68.
22. Zhongqi Lu, Zhicheng Dou, Jianxun Lian, Xing Xie, and Qiang Yang. 2015. Content-Based Collaborative Filtering for News Topic Recommendation.. In *AAAI*. Citeseer, 217–223.
23. Yasuko Matsubara, Yasushi Sakurai, Christos Faloutsos, Tomoharu Iwata, and Masatoshi Yoshikawa. 2012. Fast mining and forecasting of complex time-stamped events. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 271–279.
24. Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. 2010. Transfer Learning in Collaborative Filtering for Sparsity Reduction.. In *AAAI*, Vol. 10. 230–235.
25. Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. 2009. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 727–736.

26. Mark Rogers, Lei Li, and Stuart J Russell. 2013. Multilinear dynamical systems for tensor time series. In *Advances in Neural Information Processing Systems*. 2634–2642.

27. Adam Sadilek and John Krumm. 2012. Far out: predicting long-term human mobility. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI Press, 814–820.

28. Choonsung Shin, Jin-Hyuk Hong, and Anind K Dey. 2012. Understanding and prediction of mobile application usage for smart phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. ACM, 173–182.

29. Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. 2010. Limits of predictability in human mobility. *Science* 327, 5968 (2010), 1018–1021.

30. Yu Sun, Nicholas Jing Yuan, Yingzi Wang, Xing Xie, Kieran McDonald, and Rui Zhang. 2016a. Contextual Intent Tracking for Personal Assistants. In *Proceedings of the 22th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

31. Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. 2016b. Collaborative Nowcasting for Contextual Recommendation. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1407–1418.

32. Ye Tian, Zuoliang Ye, Yufei Yan, and Miao Sun. 2015. A practical model to predict the repeat purchasing pattern of consumers in the C2C e-commerce. *Electronic Commerce Research* 15, 4 (2015), 571–583.

33. Fang-Mei Tseng, Hsiao-Cheng Yu, and Gwo-Hsiung Tzeng. 2002. Combining neural network model with seasonal time series ARIMA model. *Technological Forecasting and Social Change* 69, 1 (2002), 71–87.

34. Dirk Van den Poel and Wouter Buckinx. 2005. Predicting online-purchasing behaviour. *European Journal of Operational Research* 166, 2 (2005), 557–575.

35. Michail Vlachos, Christopher Meek, Zografoula Vagena, and Dimitrios Gunopulos. 2004. Identifying similarities, periodicities and bursts for online search queries. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. ACM, 131–142.

36. Dashun Wang, Dino Pedreschi, Chaoming Song, Fosca Giannotti, and Albert-Laszlo Barabasi. 2011. Human mobility, social ties, and link prediction. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1100–1108.

37. Zheng Wang, Prithwish Chakraborty, Sumiko R Mekaru, John S Brownstein, Jieping Ye, and Naren Ramakrishnan. 2015. Dynamic poisson autoregression for influenza-like-illness case count prediction. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1285–1294.

38. Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. 2010. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization.. In *SDM*, Vol. 10. SIAM, 211–222.

39. Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell, and Tanzeem Choudhury. 2013. Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. In *Proceedings of the 2013 International Symposium on Wearable Computers*. ACM, 69–76.

40. Bo Yan and Guanling Chen. 2011. AppJoy: personalized mobile application discovery. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*. ACM, 113–126.

41. Nicholas Jing Yuan, Yingzi Wang, Fuzheng Zhang, Xing Xie, and Guangzhong Sun. 2013. Reconstructing individual mobility from smart card transactions: A space alignment approach. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*. IEEE, 877–886.

42. Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative Filtering Meets Mobile Recommendation: A User-Centered Approach.. In *AAAI*, Vol. 10. 236–241.