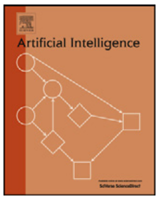
Artificial Intelligence 184–185 (2012) 17–37

Contents lists available at SciVerse ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint



Towards mobile intelligence: Learning from GPS history data for collaborative recommendation

Vincent W. Zheng [a,*], Yu Zheng [b], Xing Xie [b], Qiang Yang [a]

[a] *Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong*
[b] *Microsoft Research Asia, Building 2, No. 5 Danling Street, Haidian District, Beijing 100080, PR China*

ARTICLE INFO

Article history:
Received 3 January 2011
Received in revised form 18 February 2012
Accepted 20 February 2012
Available online 21 February 2012

Keywords:
GPS
Location
Activity
Mobile recommendation
Collaborative filtering
Personalization

ABSTRACT

With the increasing popularity of location-based services, we have accumulated a lot of location data on the Web. In this paper, we are interested in answering two popular location-related queries in our daily life: (1) if we want to do something such as sightseeing or dining in a large city like Beijing, where should we go? (2) If we want to visit a place such as the Bird's Nest in Beijing Olympic park, what can we do there? We develop a mobile recommendation system to answer these queries. In our system, we first model the users' location and activity histories as a user–location–activity rating tensor.[1] Because each user has limited data, the resulting rating tensor is essentially very sparse. This makes our recommendation task difficult. In order to address this data sparsity problem, we propose three algorithms[2] based on collaborative filtering. The first algorithm merges all the users' data together, and uses a collective matrix factorization model to provide general recommendation (Zheng et al., 2010 [3]). The second algorithm treats each user differently and uses a collective tensor and matrix factorization model to provide personalized recommendation (Zheng et al., 2010 [4]). The third algorithm is a new algorithm which further improves our previous two algorithms by using a ranking-based collective tensor and matrix factorization model. Instead of trying to predict the missing entry values as accurately as possible, it focuses on directly optimizing the ranking loss w.r.t. user preferences on the locations and activities. Therefore, it is more consistent with our ultimate goal of ranking locations/activities for recommendations. For these three algorithms, we also exploit some additional information, such as user–user similarities, location features, activity–activity correlations and user–location preferences, to help the CF tasks. We extensively evaluate our algorithms using a real-world GPS dataset collected by 119 users over 2.5 years. We show that all our three algorithms can consistently outperform the competing baselines, and our newly proposed third algorithm can also outperform our other two previous algorithms.

1. Introduction

As mobile devices with positioning functions, such as GPS-phones, become more and more popular, people are now able to find locations more easily. Based on these location data, various location-based services (LBS) are provided on the Web

* Corresponding author.
E-mail addresses: vincentz@cse.ust.hk (V.W. Zheng), yuzheng@microsoft.com (Y. Zheng), xing.xie@microsoft.com (X. Xie), qyang@cse.ust.hk (Q. Yang).

[1] A "tensor" is a multi-dimensional array (Symeonidis et al., 2008 [1]; Cichocki et al., 2009 [2]).

[2] This work is an extension to our previous work (Zheng et al., 2010 [3,4]). We propose a new model in Section 5.3 and completely re-conduct the experiments for all our three algorithms.

0004-3702/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.artint.2012.02.002

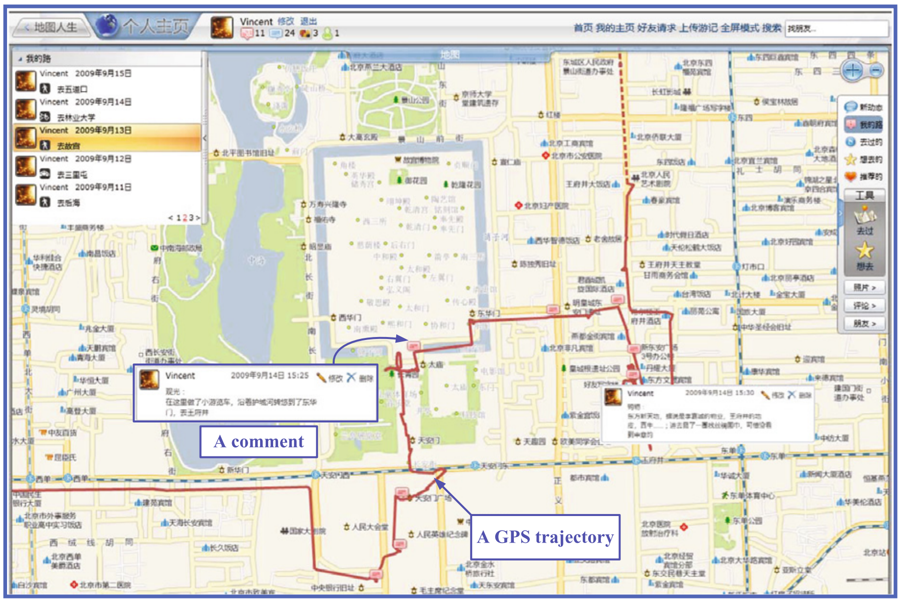


Fig. 1. GPS data management services. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

and shown to be quite attractive to users [5–7]. People can now share on the Web not only their raw GPS coordinates and time stamps, for example, for cycling route exchange,[3] but also rich text such as comments and pictures related to their trip trajectories for social blogging. In Fig. 1, we show such a location data management service from GeoLife [8], which allows users to share annotated GPS trajectories on the Web. Consider one example from this figure: after traveling around the Forbidden City in Beijing, a user tries to share some travel experiences on the Web. He then uploads his GPS trajectory of this trip, and also annotates it by attaching some interesting comments[4] (depicted as small pink boxes, each unfolding as a text box) about what he was doing, what he saw or how he felt about the places, and other useful information. Hopefully, such comments bring rich semantics to GPS trajectories and make it easier for mobile users to share their travel experiences. We expect to take such partially annotated GPS location data from many mobile users as input, and extract useful knowledge about the locations and user activities. For example, which locations are popular, and what activities are suitable at some places? Our goal is to utilize crowd wisdom encoded in their location histories to provide useful mobile recommendations. In particular, we are interested in collaborative location and activity recommendations, which are able to give both location recommendations with some activity query and activity recommendation with some location query. Here, “activity” can refer to various human behaviors such as dining, shopping, watching movies/shows, enjoying sports/exercises, tourism, and the like.

To accomplish this collaborative recommendation task, we extract location and activity information from the GPS history data for each user and formulate the recommendation problem as a collaborative filtering problem on the user–location–activity data input. We propose three collaborative filtering (CF) algorithms that rely on collective tensor and/or matrix factorization to address the data sparsity problem in recommendation:

1. A *collaborative location and activity filtering* (*CLAF*) *algorithm* [3], which merges all the users' data together, and uses a collective matrix factorization model to provide general recommendations.
2. A *personalized collaborative location and activity filtering* (*PCLAF*) *algorithm* [4], which treats each user differently and uses a collective tensor and matrix factorization to provide personalized recommendations.
3. A *ranking-based personalized collaborative location and activity filtering* (*RPCLAF*) *algorithm*, which formulates each users' pairwise preferences on the locations/activities and uses a ranking-based collective tensor and matrix factorization model to provide personalized recommendations.

We extract some auxiliary information to help the CF tasks. Such information includes the location features from the POI (points of interest) database, the activity–activity correlations from the Web, the user–user similarities from the user demographics database and the user–location preferences from the GPS trajectory data. We show that our algorithms can naturally transfer knowledge from this auxiliary information to help prediction in the target domain where location–activity rating data are sparse for the users. Among our three algorithms, the first two (i.e. *CLAF* and *PCLAF*) use square loss as

[3] http://www.bikely.com/.
[4] We consider using picture information as our future work.

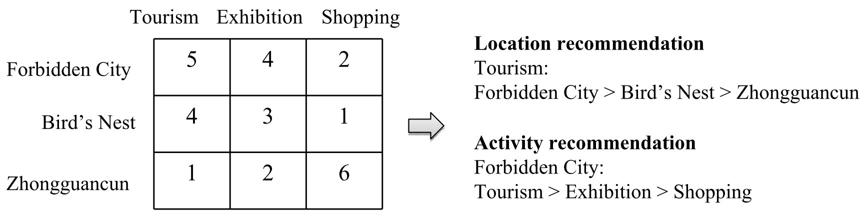


Fig. 2. Illustration of location and activity recommendation.

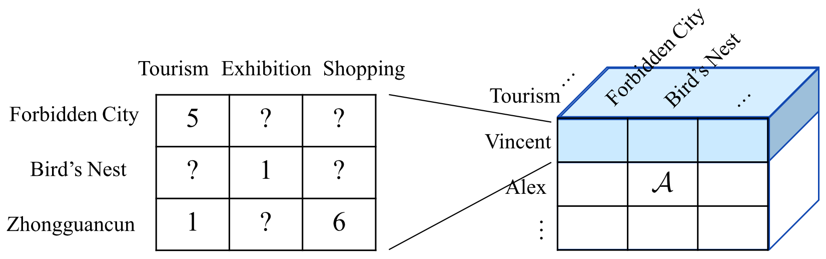


Fig. 3. Missing values in the user–location–activity tensor $\mathcal{A}$.

the optimization criteria. Specifically, their models aim to generate user–location–activity rating predictions that are as similar as the ground truth values. Then, they use their predictions to rank the locations/activities for recommendation. Different from these two algorithms, our third algorithm *RPCLAF* uses ranking loss for optimization and tries to model the user pairwise preferences on the locations/activities. The reason behind is because using ranking loss is essentially more consistent with the ultimate goal of ranking locations/activities for recommendation. Besides, it can also benefit from modeling more information with the same amount of parameters as the *PCLAF* algorithm. We use stochastic gradient descent to solve the optimization problems in our algorithms, and fill all the missing values of the user–location–activity tensor with reasonable predictions. Based on ranking over the filled location–activity matrix for each user (i.e. some slice of the user–location–activity tensor), we can provide both location recommendations and activity recommendations. Finally, we evaluate our system using a real-world GPS dataset, which was collected by 119 users over 2.5 years. The number of GPS points is around 4 million and a total distance of over 139,310 kilometers.

2. Problem statement

From the GPS data, we can extract three entities, i.e. users, locations and activities, denoting that some user visited some place and did something there. We propose to model such user–location–activity relations in a 3-D tensor, with each dimension corresponding to an entity above. In particular, we denote such a tensor as $\mathcal{A} \in \mathbb{R}^{m \times n \times r}$, where m is the number of users, n is the number of locations and r is the number of activities. Then an entry a_{ijk} in $\mathcal{A}$ denotes the frequency of a user i visiting location j and doing activity k there. When this tensor $\mathcal{A}$ is full, for each user, we can easily extract her location–activity matrix as a slice of the tensor and based on the ratings in it to do recommendations. As shown in Fig. 2, we can see location recommendation for some given activity query as a ranking over the row entry values in some column, and activity recommendation as a ranking over the column entry values in some row.

As we can see above, the recommendation is based on rankings over the complete location–activity matrix. However, in practice, the location–activity matrix for each user can be sparse due to limited number of annotations. Therefore, we may expect to have many missing entries in each user's location–activity matrix as shown in Fig. 3. Our job is to build some model which can predict a reasonable ranking on these missing entries based on what we have known with the existing entries in the tensor $\mathcal{A}$.

There are several ways to accomplish our job. A general idea is to first predict the values of such missing entries, and then based on the predicted values to give rankings for location and activity recommendation. As each user has limited location–activity ratings, it is natural to consider merging all the users' ratings together in order to get a denser location–activity matrix. Collaborative filtering can then be used to fill the existing missing entries in the matrix. Our first algorithm *CLAF* is based on such an intuition. It also exploits the location features and activity correlations as auxiliary information (as discussed later) to further alleviate the data sparsity problem. It relies on a collective matrix factorization model to fulfill the goal of collaborative filtering with the auxiliary information sources. Our *CLAF* algorithm is shown to work well in practice [3], but it is limited to provide only general recommendations. In order to provide personalized location and activity recommendations, we propose the second algorithm *PCLAF*, which directly models the users and employs a user–location–activity tensor for CF [4]. It also uses the auxiliary location and activity information, together with the user similarities and user–location preferences. Finally, it relies on a collective tensor and matrix factorization model to solve the CF problem. Our third algorithm tries to solve the CF problem from a different perspective. Considering that recommendation task is

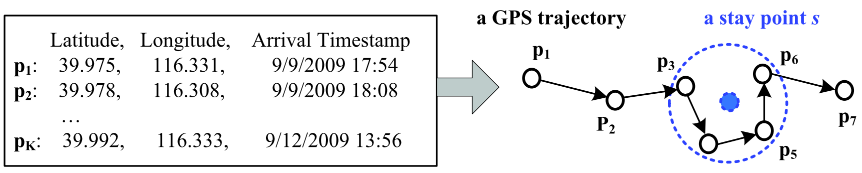

Latitude, Longitude, Arrival Timestamp
p1: 39.975, 116.331, 9/9/2009 17:54
p2: 39.978, 116.308, 9/9/2009 18:08
…
pK: 39.992, 116.333, 9/12/2009 13:56
a GPS trajectory
a stay point s
p1
P2
p3
p5
p6
p7

Fig. 4. GPS trajectory and stay point.

essentially a ranking problem, the previous two algorithms' trying to first predict the missing values and later rank them for recommendation can be taking an indirect route to solve the problem. Therefore, we can develop a model that directly uses ranking loss as the objective function, and only focuses on finding the users' pairwise preferences among locations/activities. Our newly proposed *RPCLAF* algorithm offers such a model. It uses ranking-based collective tensor and matrix factorization to incorporate the auxiliary information, and is shown to be better in ranking performance since its objective function is more consistent with ranking.

In general, there are two main categories of CF techniques. One is memory-based, using the rating data to measure the similarity between the interested matrix entities [9]. Then, these similarity values are employed to produce a prediction with a weighted average of the existing ratings. The other is model-based, relying on matrix factorization to uncover latent factors that explain observed ratings. Then, the latent factors are used to reconstruct the incomplete matrix and thus produce the rating predictions [10]. All of our three CF algorithms are model-based.

3. Overview of our system

In this section, we first clarify some terms used in this paper. Then, we discuss the application scenarios and the architecture of our system.

3.1. Preliminary

First, we clarify some terms, including GPS trajectory (*Traj*), stay point (s) and stay region (r).

Definition 1 *(GPS trajectory).* A user's trajectory *Traj* is a sequence of time-stamped points: $Traj = \langle p_0, p_1, \ldots, p_k\rangle$, where a GPS point $p_i = (x_i, y_i, t_i)$, $\forall 0 \leqslant i < k$, with t_i as a timestamp ($t_i < t_{i+1}$), and (x_i, y_i) as the two-dimension coordinates [11]. In the right part of Fig. 4, we show a trajectory consisted of 7 GPS points.

Definition 2 *(Stay point).* A stay point s stands for a geographical region where a user stayed over a time threshold T_r within a distance threshold of D_r. Denote $Dist(p_i, p_j)$ as the geospatial distance between two points p_i and p_j, and $Int(p_i, p_j) = |p_i.t_i - p_j.t_j|$ as their time interval. In a user's trajectory, s can be seen as a virtual location characterized by a set of consecutive GPS points $P = \langle p_m, p_{m+1}, \ldots, p_n\rangle$, where $\forall m < i \leqslant n$, $Dist(p_m, p_i) \leqslant D_r$, $Dist(p_m, p_{n+1}) > D_r$ and $Int(p_m, p_n) \geqslant T_r$. Hence, a stay point $s = (x, y, t_a, t_l)$, where

$$s.x = \sum_{i=m}^{n} p_i.x/|P|, \qquad s.y = \sum_{i=m}^{n} p_i.y/|P| \tag{1}$$

respectively stands for the average x and y coordinates of the collection P; $s.t_a = p_m.t_m$ is the user's arriving time on s and $s.t_l = p_n.t_n$ represents the user's leaving time [11].

Compared with raw GPS points, stay points are more meaningful in representing the locations a user stays by capturing the time duration and vicinity information, and they are commonly used as the basic units in representing the GPS data [11,12]. However, in practice, when we consider many GPS trajectories together, we may find that some stay points refer to the same interested region. This is because the users can stay in different parts (e.g. the west and east wings) of an interested region (e.g. Bird's Nest stadium). In the recommendation, we focus on a whole region of interest such as the Bird's Nest rather than its two wings, so we need to further extract some geographical regions by clustering the nearby stay points. We call these stay regions.

Definition 3 *(Stay region (location)).* Given all the stay points extracted from the GPS data as $S = \{s_1, s_2, \ldots, s_N\}$ and a clustering algorithm $Alg(S)$ taking S as input, we have a stay region r as a geographic region which contains a set of stay points $S' = \{s'_m, s'_{m+1}, \ldots, s'_n \mid s'_i \in S, \forall m \leqslant i \leqslant n\}$ belonging to some same cluster. Hence, a stay region $r = (x, y)$, where

$$sr.x = \sum_{i=m}^{n} s_i.x/\left|S'\right|, \qquad sr.y = \sum_{i=m}^{n} s_i.y/\left|S'\right| \tag{2}$$

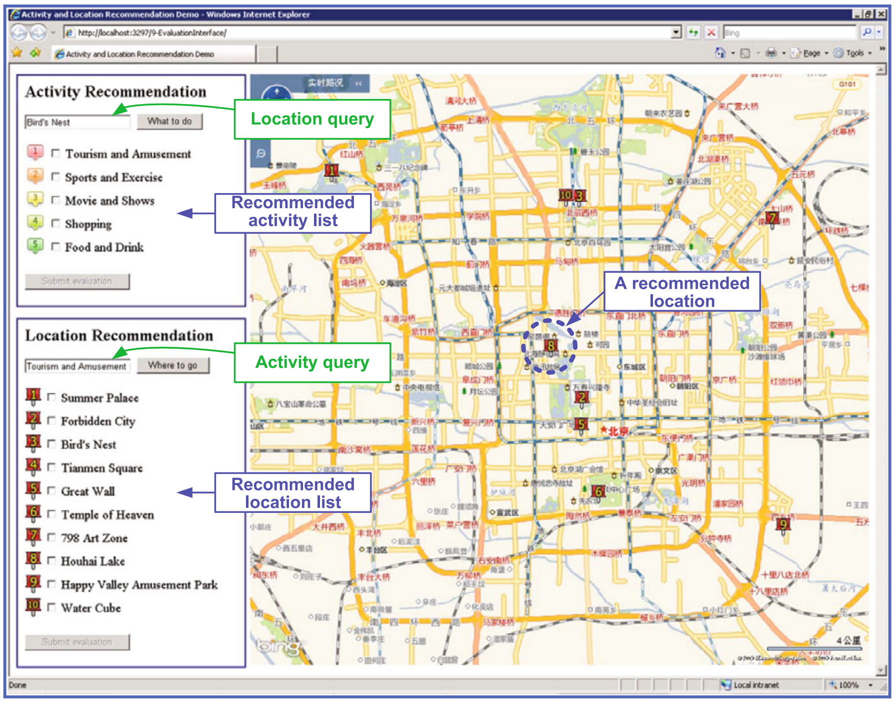


Fig. 5. User interface for our system.

stand for the average x and y coordinates of the collection S. In this work, stay regions are used as the basic units for location recommendation, i.e. when we recommend locations, in fact we recommend stay regions.

We instantiate *Alg* as a grid-based clustering algorithm in [3]. The basic idea is to divide the map into grids, and employ a greedy algorithm to iteratively assign the grid with maximal number of stay points and its neighboring grids into the same cluster. Notice that we do not directly extract stay regions by clustering the raw GPS points from all the trajectories. This is because we may lose sequential information by mixing the raw GPS points from different trajectories together, and thus it is hard to detect any meaningful stays. Interested readers may refer to our previous work [3] for more details. Compared with previous clustering algorithms such as the classic k-means algorithm and the density-based OPTICS clustering algorithm [13] that do not constrain the output cluster sizes, our grid-based clustering algorithm can make sure that the recommended locations are not be too large in size for users to find the destinations. However, we do not argue that the stay regions found by our grid-based clustering algorithm are definitely better than those found by some other clustering algorithms in terms of some other metrics like cluster coherence or density awareness.

3.2. Application scenarios and architecture

The work reported in this paper is an important component of our GeoLife project, whose prototype has been internally accessible within Microsoft since Oct. 2007. So far, we have had 119 individuals using this system.

Fig. 5 shows our system's user interface. It's organized as a Website (similar to a search engine) so that both PCs and hand-held devices can access it. To use our system, a user can choose to log in the system to get personalized recommendations or stay non-login to get general recommendations. Then, for activity recommendation, the user can input a location, such as "Bird's Nest", as a location query; then, our system can show the queried location on the map and suggest a ranking list of activities (top five here). The user can provide some feedback about the results by giving some ratings. For location recommendation, the user can input an activity, such as "tourism and amusement", as an activity query; then our system can suggest a ranking list of candidate locations (top ten here) and display them on the map, so that the user can zoom in on the map and get more details (e.g. transportation). The user can also view the location candidates ranked lower than ten to get more recommendations. Similarly, the user can also provide feedbacks on location recommendation.

For system architecture, in the back-end, our recommendation system consists several parts. First, it takes raw GPS data as input and processes them to get the meaningful stay regions as interested locations for recommendation. Second, it takes the user comments as input to extract the useful activity information for each interested location. Third, it extracts the auxiliary knowledge such as user similarities, location features and activity correlations. Fourth, it trains a recommender based on some collaborative filtering algorithm we provided. In the front-end, our system provides some interface so that the users can access the recommender through internet using laptops/PCs or PDAs/smart-phones, and submit the query (i.e.

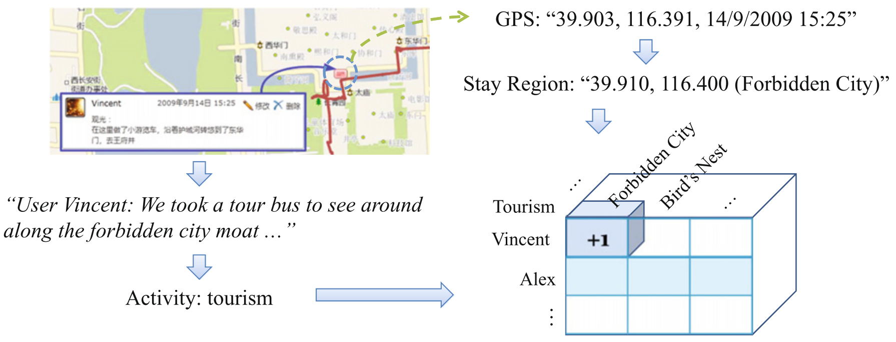

GPS: "39.903, 116.391, 14/9/2009 15:25"
Stay Region: "39.910, 116.400 (Forbidden City)"
"User Vincent: We took a tour bus to see around along the forbidden city moat ..."
Activity: tourism
Forbidden City
Bird's Nest
Tourism
Vincent
Alex
+1

Fig. 6. An example about activity information extraction.

activity or location names). Then, our system returns a ranking list of locations or activities given the activity or location query.

4. Data modeling

In this section, we introduce how to model the location–activity data for training the recommender. We also introduce how to extract auxiliary information such as location–features, activity correlations, user similarities and user–location preferences for additional inputs.

4.1. Activity information extraction

We rely on the user-generated text comments to get the user–location–activity tensor. Based on stay region extraction, we can get a set of stay regions. For each stay region sr_i in the stay region set $R = \{sr_i, 1 \leqslant i \leqslant |R|\}$, we can first extract the comments from the GPS data attached to this stay region. Consider an example shown in Fig. 6. A user visited the Forbidden City, where he attached some comment on the GPS trajectory on the map, saying that "*We took a tour bus to look around along the forbidden city moat ...*". From the GPS coordinates, we can figure out the stay region as "Forbidden City". Then, from the comment content, we can infer that the user was pursuing "Tourism". One such comment gives a rating of "1" to the location–activity pair of ⟨"Forbidden City", "Tourism"⟩. By parsing all the comments, we can count various activities on each stay region (location) for each user. Let us denote an r-dimensional count vector $\mathbf{a}_j^{(i)} = [a_{j1}^{(i)}, a_{j2}^{(i)}, \ldots, a_{jr}^{(i)}]$, where each $a_{jk}^{(i)}$ is the number of times when activity k was performed at a location j by user i. Therefore, the user–location–activity tensor $\mathcal{A}$ has its entries defined as:

$$\mathcal{A}_{ijk} = a_{jk}^{(i)}, \quad \forall i = 1, \ldots, m;\ j = 1, \ldots, n;\ k = 1, \ldots, r. \tag{3}$$

For an entry of $\mathcal{A}_{ijk} = 0$, it means that we do not observe any comment from the data indicating that user i performed activity k at location j. We treat these zero entries as missing values, in the sense that the user may still be interested in doing that activity at that location though we have not observed any indication so far.

Note that, in this study, we use human labelers to parse the user-generated comments to get the activity labels. But in general, as the user comments are basically text, one can use text classification to automatically detect the activities. For example, Nigam et al. provide an approach to use both labeled and unlabeled text data for classification, [14]. Therefore, the human labeling cost can be greatly reduced and the activity extraction becomes more scalable. We leave this as our future work.

4.2. Location-feature extraction

We use the POI category database to get the statistics (counts) of different POIs in an interested region. In particular, given a stay region $sr_i \in R$, we count the number of different POIs in an enclosing rectangle of the stay points in sr_i, with the coordinates as $[sr_i.lat - d_s/2, sr_i.lat + d_s/2] \times [sr_i.lng - d_s/2, sr_i.lng + d_s/2]$. Here, d_s is the size parameter and it is set as 500 meters in this paper. Interested readers are referred to our previous work [3] for more experimental details on this parameter. Therefore, the size of the enclosing rectangle is $d \times d$. Denote the count vector for a location j as $\mathbf{c}_j = [c_{j1}, c_{j2}, \ldots, c_{jp}]$ for p types of POIs. Consider that some types of POIs (e.g. restaurants) are more popular than others (e.g. movie theaters), we follow information retrieval to further normalize these counts in the form of term-frequency inversed-document-frequency (TF-IDF) [15] to obtain a location–feature matrix $C \in \mathbb{R}^{n \times p}$:

$$C_{jl} = \frac{c_{jl}}{\sum_{l=1}^{p} c_{jl}} \cdot \log \frac{|\{\mathbf{c}_j\}|}{|\{\mathbf{c}_j : c_{jl} > 0\}|}, \quad \forall j = 1, \ldots, n;\ l = 1, \ldots, p, \tag{4}$$

where $|\{\mathbf{c}_j\}|$ is the number of all the count vectors (i.e. number of locations), and $|\{\mathbf{c}_j\colon\ c_{jl} > 0\}|$ is the number of count vectors (i.e. locations) having non-zero l-th type POIs. In this way, we increase the weights for those important POIs that are fewer but unique (e.g. movie theaters), and decrease the weights for those extensively distributed POIs (e.g. restaurants).

4.3. Activity–activity correlation extraction

Knowing the correlation between activities can help us to better infer what the users may do in some location based on the observation of the activities performed before. One possible way to get such correlation information is to calculate it directly from the GPS data; but due to the limited number of comments, we may not get reliable results. Fortunately, such activity correlations are usually common sense and possibly reflected on the World Wide Web. To facilitate such common sense mining, we turn to Web search for help [16]. In particular, for each pair of activities, we put their names together as a query and submit it to some commercial search engine to get the Webpage hit counts. For example, given activities “food and drink” and “shopping”, we generate a query “food and drink, and shopping” and send it to Bing. Bing then returns a list of Webpages that describe these two activities together, and as expected, the number of such returned Webpages implies the correlation between them. In general, we find the hit count for “food and drink, shopping” (48.5 million[5]) is higher than that for “food and drink, and sports and exercises” (39.4 million), showing that the correlations of “food and drink” with “shopping” is higher than with “sports and exercise”, coinciding with common sense. Based on such a method, we then have an activity–activity matrix $D \in \mathbb{R}^{r\times r}$, with each entry defined as

$$D_{ij} = h_{ij}/h^*, \quad \forall i = 1, \dots, r;\ j = 1, \dots, r, \tag{5}$$

where h_{ij} is the hit count for activity i and activity j based on some search engine. In this paper, we employ a simple normalization strategy by dividing each hit count value with h^*, where $h^* = \arg\max h_{ij}$, $\forall i, j$ is the maximal hit count among all the hit counts for each pair of activities.

4.4. More information about user

In addition to the activity and location information we have extracted above, we also have the user–user matrix $B \in \mathbb{R}^{m\times m}$ which encodes the user–user similarities. In this study, we use the demographic information such as age, gender and job of each user to form a feature vector; and then, we measures the cosine similarities between each pair of users based on their demographic feature vectors. There can be some other ways to get such user–user similarities, such as using online social network services or relying on some questionnaires of each user’s friend network. But we do not exploit them here and leave them for future study. In general, we aim to use such similarity information to uncover the like-minded users in CF. Optionally, we can also extract a matrix $E \in \mathbb{R}^{m\times n}$ from the GPS data to formulate the user–location preferences. This matrix could be helpful to model the case when we only know a user visited some place but have no idea what she was doing there.

5. Mobile recommendations

Our goal is to predict a reasonable ranking on the missing entries of user–location–activity tensor $\mathcal{A}$. In addition to the existing entries in the tensor, we also have some additional inputs such as location features, activity correlations and user–user similarities that can help prediction. In the following, we propose three collaborative filtering algorithms to achieve our goal.

5.1. Collaborative location and activity filtering

As each user has limited location–activity ratings, one possible solution is to merge all the users’ ratings together in order to get a denser location–activity matrix. In particular, we can consider to compress the 3-D user–location–activity tensor into a 2-D location–activity matrix. As shown in Fig. 7, we obtain a location–activity matrix $A \in \mathbb{R}^{n\times r}$ by having $A_{ij} = \sum_{k=1}^{m} \mathcal{A}_{kij}$, $\forall i = 1, \dots, n$; $j = 1, \dots, r$. Such a matrix aggregates the ratings of all the users. Therefore, from the matrix, we can know what people usually do when they visit some place. We can use this knowledge to guide our recommendation.

Though the matrix A is already denser than the tensor $\mathcal{A}$, it still has many missing entries. Consequently, our job becomes filling the missing entries in A. For a missing entry A_{ij}, we use collaborative filtering to predict its value. In general, if we know that one location i is suitable for doing some activity j (such as “Shopping”), and another location i' is similar to location i, we may infer that location i' is also suitable for doing activity j. Such an intuition can be captured by decomposing

$$A_{ij} = \mathbf{x}_i \cdot \mathbf{y}_j,$$

[5] All the hit count values shown here are based on the search results on May 23, 2010.

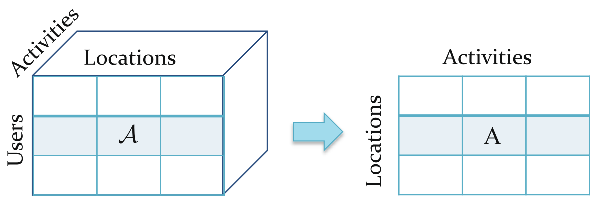


Fig. 7. Compress the user–location–activity tensor to a location–activity matrix.

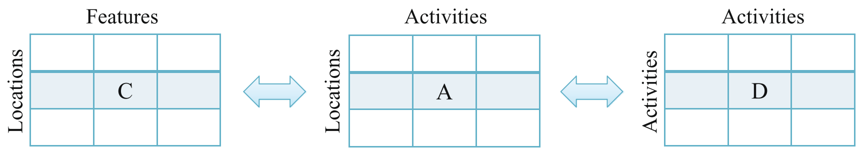


Fig. 8. Model demonstration for our *CLAF* algorithm.

where $\mathbf{x}_i \in \mathbb{R}^{n\times d}$ is the latent factor for location i and $\mathbf{y}_j \in \mathbb{R}^{r\times d}$ is the latent factor for activity j. Here, d is the latent factor dimension. The location latent factors characterize the properties of locations, and thus for similar locations i and i', their latent factors $\mathbf{x}_i$ and $\mathbf{x}_{i'}$ are similar. As a result, if location i is suitable for activity j (i.e. A_{ij} is big), then we can predict that location i' is also suitable for activity j (i.e. $A_{i'j}$ is also big), given the same activity latent factor $\mathbf{y}_j$. We can find these latent factors $\mathbf{x}_i$ and $\mathbf{y}_j$ for each location i and activity j, based on the existing entries in the matrix A. Specifically, we may try to minimize the following objective function in order to get $\mathbf{x}_i$ and $\mathbf{y}_j$:

$$\mathcal{L}(X, Y) = \sum_{(i,j)\in\mathcal{D}_A} (\mathbf{x}_i \cdot \mathbf{y}_j - A_{ij})^2, \tag{6}$$

where the loss term is computed on the existing entry set $\mathcal{D}_A$ for matrix A. In addition, we can also use location features and activity correlations to help the this optimization. For example, using the location features, we can have the prior knowledge of whether a location is similar to another location based on their feature values. Using the activity–activity correlations, we can know how likely the occurrence of one activity may imply the occurrence of another activity. One example is that, many people choose to have food and drink in the shopping mall as usually there are many restaurants and bars in/near the shopping mall. Therefore, if we observe a location is suitable for activity "shopping", it can also suitable for activity "food and drink". This information gives us some prior knowledge about the activity latent factors, and thus can help the matrix factorization of A. As shown in Fig. 8, we then aim to factorize the target location–activity matrix A, the additional location–feature matrix C and the activity–activity correlation matrix together.

Formally, we propose to employ collective matrix factorization [17] for developing a collaborative location and activity filtering model, and the objective function is:

$$\mathcal{L}(X, Y, Z) = \sum_{(i,j)\in\mathcal{D}_A} (\mathbf{x}_i \cdot \mathbf{y}_j - A_{ij})^2 + \beta_1 \sum_{(i,k)\in\mathcal{D}_C} (\mathbf{x}_i \cdot \mathbf{z}_k - C_{ik})^2$$
$$+ \beta_2 \sum_{(j,k)\in\mathcal{D}_D} D_{jk}\|\mathbf{y}_j - \mathbf{y}_k\|^2 + \beta_3\big(\|X\|_F^2 + \|Y\|_F^2 + \|Z\|_F^2\big), \tag{7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. $\beta_i \geqslant 0$, $\forall i$ are parameters to manually tune. In the above objective function, we aim to propagate the information among the matrices A, C and D, by requiring them to share some latent factors $X \in \mathbb{R}^{n\times d}$, $Y \in \mathbb{R}^{r\times d}$ and $Z \in \mathbb{R}^{p\times d}$. The first two terms in the objective function measure the loss in matrix factorization on A and C. The third term forces the learned latent activity factors $\mathbf{y}_i$ and $\mathbf{y}_j$ to be more similar if activity i and activity j have higher correlation (i.e. D_{ij} is bigger). The last term controls the regularization over the factorized matrices so as to prevent overfitting.

In general, this objective function is not jointly convex to all the variables X, Y and Z, and we cannot get closed-form solutions for minimizing the objective function. Therefore, we turn to some numerical method such as stochastic gradient descent to get the local optimal solutions. Specifically, we obtain the gradients for each variable in Table 1.

Finally, we use gradient descent to iteratively minimize the objective function, and the details are given in Algorithm 1. In each iteration, the algorithm first randomly samples one existing entry (i, j) in the matrix A by an operation `bootstrap`. Then, it updates the latent factor variables $\mathbf{x}_i$ and $\mathbf{y}_j$. It also updates all the latent factor variables $\mathbf{z}_k$'s. After having the converged X and Y, we can predict the missing values in matrix A. Based on the predictions, we can provide both location and activity recommendations. Note that, this algorithm is focused on general recommendation, so that the system gives same recommendation results to different users.

Table 1
Gradients for Eq. (7). Without loss of generality, we ignore the constant value of 1/2 throughout the whole paper's gradient derivation.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_i} = \sum_j (\mathbf{x}_i \cdot \mathbf{y}_j - A_{ij})\mathbf{y}_j + \beta_1 \sum_k (\mathbf{x}_i \cdot \mathbf{z}_k - C_{ik})\mathbf{z}_k + \beta_3 \mathbf{x}_i,$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{y}_j} = \sum_i (\mathbf{x}_i \cdot \mathbf{y}_j - A_{ij})\mathbf{x}_i + \beta_2 \sum_{k \neq j} D_{jk}(\mathbf{y}_j - \mathbf{y}_k) + \beta_3 \mathbf{y}_j,$$
$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}_k} = \beta_1 \sum_i (\mathbf{x}_i \cdot \mathbf{z}_k - C_{ik})\mathbf{x}_i + \beta_3 \mathbf{z}_k.$$

Algorithm 1 The *CLAF* algorithm

```
1: Randomly initialize the parameters $X, Y$ and $Z$;
2: repeat
3:     for $t = 1$ to $|\mathcal{D}_A|$ do
4:         $(i, j) \leftarrow \texttt{bootstrap}(\mathcal{D}_A)$;      // random sampling with replacement
5:         Update $\mathbf{x}_i \leftarrow \mathbf{x}_i - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i}$;      // according to Table 1
6:         Update $\mathbf{y}_j \leftarrow \mathbf{y}_j - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{y}_j}$;      // according to Table 1
7:         Update $\mathbf{z}_k \leftarrow \mathbf{z}_k - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{z}_k}, \forall k$;      // according to Table 1
8:     end for
9: until convergence
```

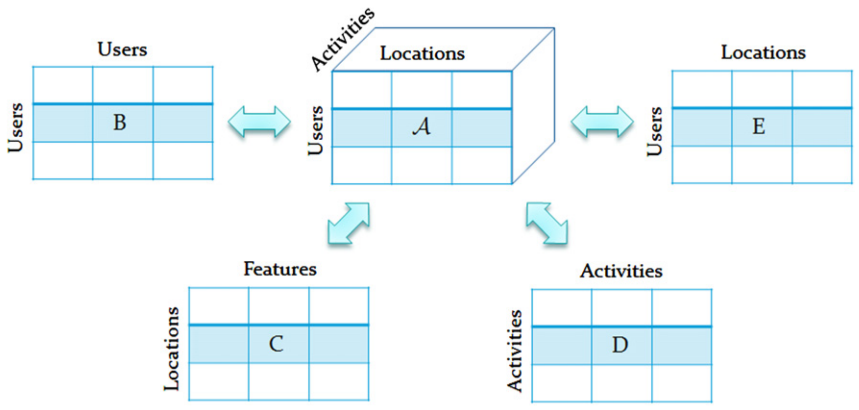


Fig. 9. Model demonstration for our *PCLAF* algorithm.

5.2. Personalized collaborative location and activity filtering

One limitation of our *CLAF* algorithm is that it cannot provide personalization to each user in recommendation. Therefore, we propose a *PCLAF* algorithm to address this problem [4]. Specifically, we directly model the user–location–activity tensor $\mathcal{A}$ under the factorization framework, and try to use as much additional information as possible to help alleviate the data sparsity issue. The model illustration for our *PCLAF* algorithm is given in Fig. 9. Our goal here is to fill the missing entries in tensor $\mathcal{A}$. In addition to the location features and activity correlations that we have used in our *CLAF* algorithm, we introduce more information for the users since we now directly model each user in collaborative filtering. In particular, we utilize the matrix $B \in \mathbb{R}^{m \times m}$ which encodes the user–user similarities. We aim to use this similarity information to uncover the like-minded users in CF. We also have another matrix $E \in \mathbb{R}^{m \times n}$ from the GPS data to model the user–location visiting preferences. It can be useful to formulate the user preferences on each location. Note that, there have been some studies on exploiting collective matrix factorization [17], or modeling the multi-dimensional (tensor) data with memory-based CF [18], or single tensor factorization [1,2], but few of them consider handling collective tensor and matrix factorization together.

To fill missing entries in the tensor $\mathcal{A}$, we follow the model-based methods [10,17] to decompose the tensor $\mathcal{A}$ w.r.t. each tensor entity (i.e. users, locations and activities). In factorization, we force the latent factors to be shared with the additional matrices so as to utilize their information. After such latent factors are obtained, we can reconstruct the tensor by filling all the missing entries. In our model, we propose a PARAFAC-style tensor factorization [2] framework to integrate the tensor with the additional matrices for regularized factorization. Specifically, our objective function is

Table 2
Gradients for Eq. (8), where $\mathcal{A}'_{ijk} = \sum_{l=1}^{d} \mathbf{x}_{il}\mathbf{y}_{jl}\mathbf{z}_{kl}$ and "$\circ$" is entry-wise product.

$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_i} = \sum_{j,k}(\mathcal{A}'_{ijk} - \mathcal{A}_{ijk})(\mathbf{y}_j \circ \mathbf{z}_k) + \lambda_1 \sum_{j \neq i} B_{ij}(\mathbf{x}_i - \mathbf{x}_j) + \lambda_4 \sum_j (\mathbf{x}_i \cdot \mathbf{y}_j - E_{ij})\mathbf{y}_j + \lambda_5 \mathbf{x}_i$
$\frac{\partial \mathcal{L}}{\partial \mathbf{y}_j} = \sum_{i,k}(\mathcal{A}'_{ijk} - \mathcal{A}_{ijk})(\mathbf{x}_i \circ \mathbf{z}_k) + \lambda_2 \sum_l (\mathbf{y}_j \cdot \mathbf{v}_l - C_{jl})\mathbf{v}_l + \lambda_4 \sum_i (\mathbf{x}_i \cdot \mathbf{y}_j - E_{ij})\mathbf{x}_i + \lambda_5 \mathbf{y}_j$
$\frac{\partial \mathcal{L}}{\partial \mathbf{z}_k} = \sum_{i,j}(\mathcal{A}'_{ijk} - \mathcal{A}_{ijk})(\mathbf{x}_i \circ \mathbf{y}_j) + \lambda_3 \sum_{l \neq k} D_{kl}(\mathbf{z}_k - \mathbf{z}_l) + \lambda_5 \mathbf{z}_k$
$\frac{\partial \mathcal{L}}{\partial \mathbf{v}_l} = \lambda_2 \sum_j (\mathbf{y}_j \cdot \mathbf{v}_l - C_{jl})\mathbf{y}_j + \lambda_5 \mathbf{v}_l$

Algorithm 2 The *PCLAF* algorithm

```
1: Randomly initialize the parameters $X, Y, Z$ and $V$;
2: repeat
3:   for $t = 1$ to $|\mathcal{D}_\mathcal{A}|$ do
4:     $(i, j, k) \leftarrow$ bootstrap($\mathcal{D}_\mathcal{A}$);   // random sampling with replacement
5:     Update $\mathbf{x}_i \leftarrow \mathbf{x}_i - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i}$;   // according to Table 2
6:     Update $\mathbf{y}_j \leftarrow \mathbf{y}_j - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{y}_j}$;   // according to Table 2
7:     Update $\mathbf{z}_k \leftarrow \mathbf{z}_k - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{z}_k}$;   // according to Table 2
8:     Update $\mathbf{v}_l \leftarrow \mathbf{v}_l - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{v}_l}$, $\forall l$;   // according to Table 2
9:   end for
10: until convergence
```

$$\mathcal{L}(X, Y, Z, V) = \sum_{(i,j,k) \in \mathcal{D}_\mathcal{A}} \left(\sum_{l=1}^{d} \mathbf{x}_{il}\mathbf{y}_{jl}\mathbf{z}_{kl} - \mathcal{A}_{ijk} \right)^2 + \lambda_1 \sum_{(i,j) \in \mathcal{D}_B} B_{ij} \|\mathbf{x}_i - \mathbf{x}_j\|^2 + \lambda_2 \sum_{(j,l) \in \mathcal{D}_C} (\mathbf{y}_j \cdot \mathbf{v}_l - C_{jl})^2$$
$$+ \lambda_3 \sum_{(k,l) \in \mathcal{D}_D} D_{kl} \|\mathbf{z}_k - \mathbf{z}_l\|^2 + \lambda_4 \sum_{(i,j) \in \mathcal{D}_E} (\mathbf{x}_i \cdot \mathbf{y}_j - E_{ij})^2 + \lambda_5 \big(\|X\|^2 + \|Y\|^2 + \|Z\|^2 + \|V\|^2 \big), \qquad (8)$$

where $X \in \mathbb{R}^{m \times d}$, $Y \in \mathbb{R}^{n \times d}$, $Z \in \mathbb{R}^{r \times d}$ and $V \in \mathbb{R}^{p \times d}$ are the matrix forms of latent factors for user, location, activity and location features, respectively. λ_1–λ_5 are model parameters; and when $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0$, our model degenerates to the standard PARAFAC tensor decomposition. This shows that our model is more flexible to utilize other information about the targeted entities. In the above objective function, the first term decomposes the user–location–activity tensor $\mathcal{A}$ as an outer-product of three latent factors w.r.t. each entity. The second term poses a regularization term on the users, forcing the latent factors of two users to be as close as possible if they are similar according to matrix B. The third term borrows the similar idea with collective matrix factorization [17], by sharing the location latent factor Y with the tensor factorization. The fourth term is a regularization term similar to the second term, forcing the latent factors of two activities to be as close as possible w.r.t. their correlations. The fifth term shares the user latent factor X and location latent factor Y with the tensor factorization. The last term is a regularization term in order to prevent overfitting.

In general, there is no closed form solution for Eq. (8), so we again use stochastic gradient descent to solve the problem. The gradients are listed in Table 2, and the algorithm details are given in Algorithm 2. After having the converged X, Y and Z, we can predict the missing values in tensor $\mathcal{A}$.

5.3. Ranking-based personalized collaborative location and activity filtering

Recall that our job is to build some model which can predict a reasonable ranking on these missing entries in tensor $\mathcal{A}$. Both our previous algorithms, *CLAF* and *PCLAF* aim to find some model that can minimize the prediction errors (e.g. in terms of square loss) w.r.t. the existing ground truth ratings. After the model is learned, predictions on the missing values are used for ranking in order to output recommendation results. Considering that in recommendation we are essentially interested in ranking results, such a learning strategy may take an indirect route to solve the problem. In this section, we propose a new algorithm, which takes a direct way to solve the recommendation problem by using ranking loss as the objective function. In particular, our new algorithm, *RPCLAF*, tries to formulate the user's pairwise preferences to different location–activity pairs. By learning with such partial rankings, our model is able to directly deliver the ranking results on missing entries.

Compared with our previous two algorithms *CLAF* and *PLCAF*, our new *RPCLAF* algorithm has several advantages. First, the objective function is based on ranking loss which is more consistent with the final goal, so the model may generate better results. Second, compared with *PLCAF* which considers each rating independently, *RPCLAF* takes rating pairs as input and thus has more data for training. Given that *RPCLAF* has the same number of latent factor variables as *PCLAF*, using more data can help improve the performance. Third, using ranking-loss is potentially useful for handling the different rating scales among the tensor and the matrices, as it on focuses on the pairwise ranking rather than the absolute values [19]. Note that, there have been some studies using ranking loss for collaborative filtering [20,21], but few of them consider it in the collective tensor and matrix factorization scenario.

First, let us define the location–activity pairwise preference as

$$\eta_{i,j,k,j',k'} = \begin{cases} +1 & \text{if } \mathcal{A}_{i,j,k} > \mathcal{A}_{i,j',k'} \mid (i,j,k) \in I_i \wedge (i,j',k') \in I_i; \\ 0 & \text{if } \mathcal{A}_{i,j,k} = \mathcal{A}_{i,j',k'} \mid (i,j,k) \in I_i \wedge (i,j',k') \in I_i; \\ -1 & \text{if } \mathcal{A}_{i,j,k} < \mathcal{A}_{i,j',k'} \mid (i,j,k) \in I_i \wedge (i,j',k') \in I_i; \\ ? & \text{if } (i,j,k) \notin I_i \vee (i,j',k') \notin I_i; \end{cases}$$

where I_i denotes the existing entries for user i in tensor $\mathcal{A}$. Then, in order to formulate the probability for these pairwise preferences, we follow the Bradley–Terry model [22,19] by defining

$$\begin{aligned} &p(\eta_{i,j,k,j',k'} = +1) = \sigma(\mathcal{A}_{i,j,k} - \mathcal{A}_{i,j',k'}), \\ &p(\eta_{i,j,k,j',k'} = -1) = \sigma(\mathcal{A}_{i,j',k'} - \mathcal{A}_{i,j,k}), \\ &p(\eta_{i,j,k,j',k'} = 0) = 1 - \sigma(\mathcal{A}_{i,j,k} - \mathcal{A}_{i,j',k'}) - \sigma(\mathcal{A}_{i,j',k'} - \mathcal{A}_{i,j,k}), \end{aligned}$$

where $\sigma(x) = \frac{1}{1+\theta e^{-x}}$ is the logistic sigmoid function. The positive parameter $\theta \geqslant 1$ controls the probability of ties.

Given the Bradley–Terry model, one can easily formulate the data loglikelihood. Specifically, denote $\mathcal{D}_{+1} = \{(i,j,k,j',k') \mid \eta_{i,j,k,j',k'} = +1\}$ as the set of data with positive preference, and $\mathcal{D}_0 = \{(i,j,k,j',k') \mid \eta_{i,j,k,j',k'} = 0\}$ as the set of data with preference ties. Therefore, we construct a pairwise preference data set $\mathcal{D}_{\mathcal{A}} = \mathcal{D}_{+1} \cup \mathcal{D}_0$. The loss function is the negative loglikelihood:

$$\mathcal{L}_{tensor} = -\sum_{(i,j,k,j',k') \in \mathcal{D}_{+1}} \ln p(\eta_{i,j,k,j',k'} = +1) - \sum_{(i,j,k,j',k') \in \mathcal{D}_0} \ln p(\eta_{i,j,k,j',k'} = 0), \tag{9}$$

where the tensor rating is factorized in a PARAFAC manner as our *PCLAF* algorithm: $\mathcal{A}_{i,j,k} \triangleq \sum_{l=1}^{d} \mathbf{x}_{il}\mathbf{y}_{jl}\mathbf{z}_{kl}$.

Similar to our previous *PCLAF* algorithm, we also utilize the user–location matrix to model user preferences on locations. In particular, we define the pairwise preference as:

$$\zeta_{u,j,j'} = \begin{cases} +1 & \text{if } E_{i,j} > E_{i,j'} \mid (i,j) \in J_i \wedge (i,j') \in J_i \wedge (i,j,\cdot,j',\cdot) \in \mathcal{D}_{\mathcal{A}}; \\ 0 & \text{if } E_{i,j} = E_{i,j'} \mid (i,j) \in J_i \wedge (i,j') \in J_i \wedge (i,j,\cdot,j',\cdot) \in \mathcal{D}_{\mathcal{A}}; \\ -1 & \text{if } E_{i,j} < E_{i,j'} \mid (i,j) \in J_i \wedge (i,j') \in J_i \wedge (i,j,\cdot,j',\cdot) \in \mathcal{D}_{\mathcal{A}}; \\ ? & \text{otherwise.} \end{cases}$$

J_i is the set of existing entries for user i in matrix E. Denote $\mathcal{D}'_{+1} = \{(i,j,j') \mid \zeta_{i,j,j'} = +1\}$ as the set of data with positive preference, $\mathcal{D}'_{-1} = \{(i,j,j') \mid \zeta_{i,j,j'} = -1\}$ as the set of data with negative preference and $\mathcal{D}'_0 = \{(i,j,j') \mid \zeta_{i,j,j'} = 0\}$ as the set of data with tied preference. Therefore, the negative loglikelihood is

$$\mathcal{L}_{pref} = -\lambda_4 \Bigg[\sum_{(i,j,j') \in \mathcal{D}'_{+1}} \ln p(\zeta_{i,j,j'} = +1) + \sum_{(i,j,j') \in \mathcal{D}'_{-1}} \ln p(\zeta_{i,j,j'} = -1) + \sum_{(i,j,j') \in \mathcal{D}'_0} \ln p(\zeta_{i,j,j'} = 0)\Bigg], \tag{10}$$

where the matrix rating is factorized as $E_{i,j} \triangleq \mathbf{x}_i \cdot \mathbf{y}_j$.

For the other auxiliary information such as user similarities, location features and activity correlations, we define the loss function as

$$\mathcal{L}_{aux} = \lambda_1 \sum_{(i,l) \in \mathcal{D}_B} B_{il} \|\mathbf{x}_i - \mathbf{x}_l\|^2 + \lambda_2 \sum_{(j,l) \in \mathcal{D}_C} (\mathbf{y}_j \cdot \mathbf{v}_l - C_{jl})^2 + \lambda_3 \sum_{(k,l) \in \mathcal{D}_D} D_{kl} \|\mathbf{z}_k - \mathbf{z}_l\|^2, \tag{11}$$

and the regularization term is

$$\mathcal{R} = \lambda_5 \big(\|X\|^2 + \|Y\|^2 + \|Z\|^2 + \|V\|^2\big), \tag{12}$$

where all the λ's are positive real numbers.

Finally, we aim to minimize the following objective function

$$\mathcal{L}(X,Y,Z,V) = \mathcal{L}_{tensor} + \mathcal{L}_{aux} + \mathcal{L}_{pref} + \mathcal{R}. \tag{13}$$

We use stochastic gradient descent to solve this minimization problem, and calculate the gradients for each parameter as shown in Tables 3, 4 and 5. The algorithm details are given in Algorithm 3. In each iteration, the algorithm first randomly samples one user i and her two existing rating entries (i,j,k), (i,j',k') entries in the tensor $\mathcal{A}$ by an operation `bootstrap`.

Table 3
Gradients for Eq. (9).

$\frac{\partial \mathcal{L}_{tensor}}{\partial \mathbf{x}_i} = \begin{cases} (\sigma_{ijkj'k'} - 1)(\mathbf{y}_j \circ \mathbf{z}_k - \mathbf{y}_{j'} \circ \mathbf{z}_{k'}) & \text{if } (i,j,k,j',k') \in \mathcal{D}_{+1}; \\ (\sigma_{ijkj'k'} - \sigma_{ij'k'jk})(\mathbf{y}_j \circ \mathbf{z}_k - \mathbf{y}_{j'} \circ \mathbf{z}_{k'}) & \text{if } (i,j,k,j',k') \in \mathcal{D}_0. \end{cases}$
$\frac{\partial \mathcal{L}_{tensor}}{\partial \mathbf{y}_j} = \begin{cases} (\sigma_{ijkj'k'} - 1)(\mathbf{x}_i \circ \mathbf{z}_k) & \text{if } (i,j,k,j',k') \in \mathcal{D}_{+1}; \\ (\sigma_{ijkj'k'} - \sigma_{ij'k'jk})(\mathbf{x}_i \circ \mathbf{z}_k) & \text{if } (i,j,k,j',k') \in \mathcal{D}_0. \end{cases}$
$\frac{\partial \mathcal{L}_{tensor}}{\partial \mathbf{y}_{j'}} = \begin{cases} (\sigma_{ijkj'k'} - 1)(-\mathbf{x}_i \circ \mathbf{z}_{k'}) & \text{if } (i,j,k,j',k') \in \mathcal{D}_{+1}; \\ (\sigma_{ijkj'k'} - \sigma_{ij'k'jk})(-\mathbf{x}_i \circ \mathbf{z}_{k'}) & \text{if } (i,j,k,j',k') \in \mathcal{D}_0. \end{cases}$
$\frac{\partial \mathcal{L}_{tensor}}{\partial \mathbf{z}_k} = \begin{cases} (\sigma_{ijkj'k'} - 1)(\mathbf{x}_i \circ \mathbf{y}_j) & \text{if } (i,j,k,j',k') \in \mathcal{D}_{+1}; \\ (\sigma_{ijkj'k'} - \sigma_{ij'k'jk})(\mathbf{x}_i \circ \mathbf{y}_j) & \text{if } (i,j,k,j',k') \in \mathcal{D}_0. \end{cases}$
$\frac{\partial \mathcal{L}_{tensor}}{\partial \mathbf{z}_{k'}} = \begin{cases} (\sigma_{ijkj'k'} - 1)(-\mathbf{x}_i \circ \mathbf{y}_{j'}) & \text{if } (i,j,k,j',k') \in \mathcal{D}_{+1}; \\ (\sigma_{ijkj'k'} - \sigma_{ij'k'jk})(-\mathbf{x}_i \circ \mathbf{y}_{j'}) & \text{if } (i,j,k,j',k') \in \mathcal{D}_0. \end{cases}$

Table 4
Gradients for Eq. (10).

$\frac{\partial \mathcal{L}_{pref}}{\partial \mathbf{x}_i} = \begin{cases} \lambda_4(\sigma_{ijj'} - 1)(\mathbf{y}_j - \mathbf{y}_{j'}) & \text{if } (i,j,j') \in \mathcal{D}'_{+1}; \\ \lambda_4(\sigma_{ijj'} - \sigma_{ij'j})(\mathbf{y}_j - \mathbf{y}_{j'}) & \text{if } (i,j,j') \in \mathcal{D}'_0; \\ \lambda_4(1 - \sigma_{ij'j})(\mathbf{y}_j - \mathbf{y}_{j'}) & \text{if } (i,j,j') \in \mathcal{D}'_{-1}; \end{cases}$
$\frac{\partial \mathcal{L}_{pref}}{\partial \mathbf{y}_j} = \begin{cases} \lambda_4(\sigma_{ijj'} - 1)\mathbf{x}_i & \text{if } (i,j,j') \in \mathcal{D}'_{+1}; \\ \lambda_4(\sigma_{ijj'} - \sigma_{ij'j})\mathbf{x}_i & \text{if } (i,j,j') \in \mathcal{D}'_0; \\ \lambda_4(1 - \sigma_{ij'j})\mathbf{x}_i & \text{if } (i,j,j') \in \mathcal{D}'_{-1}; \end{cases}$
$\frac{\partial \mathcal{L}_{pref}}{\partial \mathbf{y}_{j'}} = \begin{cases} \lambda_4(\sigma_{ijj'} - 1)(-\mathbf{x}_i) & \text{if } (i,j,j') \in \mathcal{D}'_{+1}; \\ \lambda_4(\sigma_{ijj'} - \sigma_{ij'j})(-\mathbf{x}_i) & \text{if } (i,j,j') \in \mathcal{D}'_0; \\ \lambda_4(1 - \sigma_{ij'j})(-\mathbf{x}_i) & \text{if } (i,j,j') \in \mathcal{D}'_{-1}. \end{cases}$

Table 5
Gradients for Eqs. (11) and (12).

$\frac{\partial \mathcal{L}_{aux}}{\partial \mathbf{x}_i} = \lambda_1 \sum_{l \neq i} B_{i,l}(\mathbf{x}_i - \mathbf{x}_l)$	$\frac{\partial \mathcal{R}}{\partial \mathbf{x}_i} = \lambda_5 \mathbf{x}_i$
$\frac{\partial \mathcal{L}_{aux}}{\partial \mathbf{y}_j} = \lambda_2 \sum_l (\mathbf{y}_j \cdot \mathbf{v}_l - C_{jl})\mathbf{v}_l$	$\frac{\partial \mathcal{R}}{\partial \mathbf{y}_l} = \lambda_5 \mathbf{y}_j$
$\frac{\partial \mathcal{L}_{aux}}{\partial \mathbf{y}_{j'}} = \lambda_2 \sum_l (\mathbf{y}_{j'} \cdot \mathbf{v}_l - C_{j'l})\mathbf{v}_l$	$\frac{\partial \mathcal{R}}{\partial \mathbf{y}_{l'}} = \lambda_5 \mathbf{y}_{j'}$
$\frac{\partial \mathcal{L}_{aux}}{\partial \mathbf{z}_k} = \lambda_3 \sum_{l \neq k} D_{kl}(\mathbf{z}_k - \mathbf{z}_l),$	$\frac{\partial \mathcal{R}}{\partial \mathbf{z}_a} = \lambda_5 \mathbf{z}_k$
$\frac{\partial \mathcal{L}_{aux}}{\partial \mathbf{z}_{k'}} = \lambda_3 \sum_{l \neq k'} D_{k'l}(\mathbf{z}_{k'} - \mathbf{z}_l)$	$\frac{\partial \mathcal{R}}{\partial \mathbf{z}_{a'}} = \lambda_5 \mathbf{z}_{k'}$
$\frac{\partial \mathcal{L}_{aux}}{\partial \mathbf{v}_l} = \lambda_2 \sum_j (\mathbf{y}_j \cdot \mathbf{v}_l - C_{jl})\mathbf{y}_j$	$\frac{\partial \mathcal{R}}{\partial \mathbf{v}_i} = \lambda_5 \mathbf{v}_l$

Then, based on the partial ranking on $\mathcal{A}_{ijk}$ and $\mathcal{A}_{ij'k'}$, it considers different $\eta_{i,j,k,j',k'}$'s to derive the gradients on tensor loss. Given the sampled (i, j, j'), the algorithm considers different $\zeta_{i,j,j'}$'s to derive the gradients on user–location preference loss. The gradients on auxiliary information and regularization terms are further calculated. Then, the algorithm updates the latent factor variables $\mathbf{x}_i$, $\mathbf{y}_j$ and $\mathbf{z}_k$. It also updates all the latent factor variables $\mathbf{v}_l$'s. After having the converged X, Y and Z, we can predict the missing values in tensor $\mathcal{A}$ for ranking. Note that, compared with the previous *PCLAF* algorithm, our current *RPCLAF* can benefit from modeling more data (i.e. entry pairs rather than just each entry) without increasing the number of model parameters.

6. Experimental setup on real-world data

6.1. GPS users, devices and data

In our experiments, we got data from 119 users who carried GPS devices to record their outdoor trajectories from April 2007 to Oct. 2009. Fig. 10(a) shows the GPS devices used to collect data, which are comprised of stand-alone GPS receivers and GPS phones. In general, the sampling rate for GPS devices was set as two seconds. The GPS logs were collected in China, as well as a few cities in the United States, South Korea, and Japan. As most parts of the logs were generated in Beijing, and for easier evaluation of our system, we extract the logs from Beijing for our experiments. After this data preprocessing, we obtain a dataset having around 13,000 GPS trajectories with a total of around 4,000,000 GPS points and a total trajectory

Algorithm 3 The *RPCLAF* algorithm

```
1: Randomly initialize the parameters $X, Y, Z$ and $V$;
2: repeat
3:     for $t = 1$ to $|\mathcal{D}_{\mathcal{A}}|$ do
4:         $(i, j, k, j', k') \leftarrow \texttt{bootstrap}(\mathcal{D}_{\mathcal{A}})$;
5:         Update $\mathbf{x}_i \leftarrow \mathbf{x}_i - \gamma \frac{\partial \mathcal{L}}{\partial \mathbf{x}_i} = \mathbf{x}_i - \gamma \left( \frac{\partial \mathcal{L}_{tensor}}{\partial \mathbf{x}_i} + \frac{\partial \mathcal{L}_{pref}}{\partial \mathbf{x}_i} + \frac{\partial \mathcal{L}_{aux}}{\partial \mathbf{x}_i} + \frac{\partial \mathcal{R}}{\partial \mathbf{x}_i} \right)$;
6:         Similarly, update $\mathbf{y}_l \leftarrow \mathbf{y}_j - \gamma \left( \frac{\partial \mathcal{L}}{\partial \mathbf{y}_j} \right)$, $\mathbf{y}_{j'} \leftarrow \mathbf{y}_{j'} - \gamma \left( \frac{\partial \mathcal{L}}{\partial \mathbf{y}_{j'}} \right)$, $\mathbf{z}_k \leftarrow \mathbf{z}_k - \gamma \left( \frac{\partial \mathcal{L}}{\partial \mathbf{z}_k} \right)$, $\mathbf{z}_{k'} \leftarrow \mathbf{z}_{k'} - \gamma \left( \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{k'}} \right)$;
7:         Update $\mathbf{v}_l \leftarrow \mathbf{v}_l - \gamma \left( \frac{\partial \mathcal{L}}{\partial \mathbf{v}_l} \right)$, $\forall l$;
8:     end for
9: until convergence
end
```

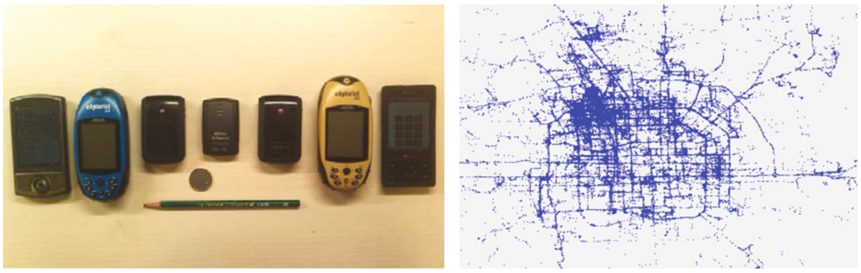

(a) GPS devices. (b) Data distribution in Beijing.

Fig. 10. GPS devices and data distribution.

Table 6
Activities that we used in the experiments.

Activities	Descriptions
Food and drink	Dinning/drinking at restaurants/bars, etc.
Shopping	Supermarkets, department stores, etc.
Movie and shows	Movie/shows in theaters and exhibition in museums, etc.
Sports and exercise	Doing exercises at stadiums, parks, etc.
Tourism and amusement	Tourism, amusement park, etc.

length of around 139,000 kilometers. To make sure that we recommend useful locations and activities, we also remove some GPS points for work and home. The data distribution in Beijing is shown in Fig. 10(b). To protect the users' privacy, we use these data anonymously.

In this study, we first extract the stay regions with our grid-based clustering algorithm. These stay regions, or locations, have a limited size of 500 meters $\times$ 500 meters, and at least 12 stay point records. In order to correctly evaluate our models, we remove all the users/locations/activities without comments. After processing, we have 119 users, 68 locations and five activities. Specifically, the five activities are defined in Table 6. We gather more user comments, and in this study, each user has 8.9 comments on average. As a user may have multiple activities in one location at a time, a comment can bring more than one ratings. After processing, on average, each user has 11.7 ratings (i.e. 11.7 entries with values) for her location–activity matrix. In our experiments, we (randomly) split some percentage of these known ratings for training and the other as the hold-out set for testing. We do not use any unknown entry in evaluation.

6.2. Evaluation methodology

We employ an objective evaluation methodology to evaluate our algorithms. Specifically, at each trial, we randomly split some percentage (e.g. 30%) of the existing tensor entries for training and hold out the other for testing. Then, we employ two metrics; one is RMSE (root mean square error) to measure the tensor/matrix reconstruction loss on a hold-out test data. For RMSE, the smaller, the better. The other metric is AUC (area under the ROC curve), to measure the ranking results based on the reconstructed tensor from training data.[6] Following the definition in [21], we design the AUC score for location ranking (averaged by m users) as

[6] The reason why we use AUC instead of nDCG (normalized discounted cumulative gain) is that, given that our data are very sparse, the length of rank list is usually short (e.g. around 2–3), and thus nDCG values tend to be close for all kinds of algorithms. As opposed to nDCG, AUC is more discriminative to measure the ranking over all data pairs.

$$AUC_{loc} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{r_i}\sum_{k=1}^{r_i}\frac{1}{|\mathcal{D}_{i\cdot k}|}\underbrace{\sum_{(j,j')\in\mathcal{D}_{i\cdot k}}\delta(\hat{\mathcal{A}}_{ijk},\hat{\mathcal{A}}_{ij'k})}_{\#(\text{correct orders})},$$

where r_i is the number of activities that user i has in test data. $\mathcal{D}_{i\cdot k}$ is the set of location test data pairs for user i on activity k. The indicator function $\delta(\hat{\mathcal{A}}_{ijk},\hat{\mathcal{A}}_{ij'k}) = 1$ if $(\hat{\mathcal{A}}_{ijk} - \hat{\mathcal{A}}_{ij'k})(\mathcal{A}_{ijk} - \mathcal{A}_{ij'k}) > 0$ or $(\hat{\mathcal{A}}_{ijk} - \hat{\mathcal{A}}_{ij'k}) \leqslant \epsilon \wedge (\mathcal{A}_{ijk} - \mathcal{A}_{ij'k}) = 0$. Here, ϵ is a tolerance parameter to measure the ties. We tentatively set $\epsilon = 0.1$ and later study its impact. Similarly, we have the AUC score for activity ranking (averaged by m users) as

$$AUC_{act} = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n_i}\sum_{j=1}^{n_i}\frac{1}{|\mathcal{D}_{ij\cdot}|}\underbrace{\sum_{(k,k')\in\mathcal{D}_{ij\cdot}}\delta(\hat{\mathcal{A}}_{ijk},\hat{\mathcal{A}}_{ijk'})}_{\#(\text{correct orders})}.$$

For AUC, the larger, the better. Finally, we run the experiments five times to generate the mean values and standard deviations of the results.

6.3. System performances

We compare our three algorithms (*CLAF*, *PCLAF* and *RPCLAF*) with six competing baselines, including user-based CF (UCF), location-based CF (LCF), activity-based CF (ACF), unifying user–location–activity CF (ULACF), single CF (SCF) and POI count based ranking (POIC). In this experiment, we set the model parameters $\lambda_1 = \lambda_2 = \lambda_4 = \lambda_5 = 0.1$, $\lambda_3 = 1$, $\beta_1 = \beta_3 = 0.1$, $\beta_2 = 1$, $k = 4$, $\theta = e^1$. We study the impact of these model parameters later.

Baseline algorithms. The first three baselines (i.e. UCF, LCF and ACF) are memory-based methods, adapted from [23] to consider CF on each tensor slice. In particular, for UCF, we consider CF on each user–location matrix for each activity independently. On each matrix, we follow [23] and use Pearson correlation as the user similarity weights. We find the top N similar users for some target user (with missing entries) and then compute their weighted average to predict the missing entry. Similarly, we have LCF and ACF by considering CF on each location–activity matrix for each user individually. In the experiments, we set $N = 4$ since we find that the prediction results do not vary significantly with N.

The fourth baseline, ULACF, is also a memory-based method, adapted from [9] to take both the tensor and the additional matrices into consideration. In particular, for each missing entry in the tensor, we extract a set of top N_u similar users, top N_l similar locations and top N_a similar activities. Then, we use the ratings from these users on the corresponding locations and activities in a weighted manner to calculate the entry value:

$$\hat{\mathcal{A}}_{i,j,k} = \frac{\sum_{u\in R_i} S_{u,i}\mathcal{A}_{u,j,k}}{4\sum_u S_{u,i}} + \frac{\sum_{l\in R_j} S_{l,j}\mathcal{A}_{i,l,k}}{4\sum_l S_{l,j}} + \frac{\sum_{a\in R_k} S_{a,k}\mathcal{A}_{i,j,a}}{4\sum_a S_{a,k}} + \frac{\sum_{u\in R_i,l\in R_j,a\in R_k} S_{u,l,a}\mathcal{A}_{u,l,a}}{4\sum_{u,l,a} S_{u,l,a}},$$

where $S_{u,i}$ is the similarity for users i and u learned from the user–user matrix B; $S_{l,j}$ is the similarity for locations j and l learned from the location–feature matrix C and the user–location matrix E by equally combining the cosine similarities calculated from each; $S_{a,k}$ is the similarity for activities k and a learned from activity–activity matrix D; $S_{u,l,a}$ is the similarity between $\mathcal{A}_{i,j,k}$ and $\mathcal{A}_{u,l,a}$ for some (u,l,a) in the neighboring sets R_i, R_j, R_k of user i, location j and activity k, respectively. It's designed as

$$S_{u,l,a} = 1/\sqrt{(1/S_{u,i})^2 + (1/S_{l,j})^2 + (1/S_{a,k})^2}.$$

In the experiments, we set $N_u = N_l = N_a = 4$, as similar to the previous cases.

The fifth baseline, SCF, is a model-based model employed to compare with our algorithm *CLAF* [10]. Similarly, it also takes the location–activity matrix A as input for CF. The model aims to find the latent location factor $\mathbf{x}$ and latent activity factor $\mathbf{y}$ that minimize the loss in Eq. (6).

The last baseline, POIC, is a baseline that uses the POI counts on each location to generate the ranking results for both location and activity recommendation. In particular, we count the number of POIs in each location for each activity category. Then, we normalize the counts to [0, 1] and use them to give the rankings. Generally, if a location has more restaurant and bar POIs than the other types of POIs, then it is assumed to be more suitable for activity of "food and drink", regardless of what the mobile users really do there.

Results. The comparison results are shown in Table 7. We report two settings of results here: using 30% of data for training and using 50% of data for training. As we can see in both settings, our algorithms generally outperform the baselines, showing the effectiveness of our models. Note that, as our *PCLAF*'s objective function is based on square loss (and it well integrates the other auxiliary information), it has the lowest RMSE values among all the algorithms. As opposed to square loss,

Table 7
Comparison with baselines, with different percentages of data used for training.

	RMSE		AUC_{loc}		AUC_{act}	
Percent	30%	50%	30%	50%	30%	50%
CLAF	0.35 ± 0.02	0.36 ± 0.03	0.73 ± 0.03	0.80 ± 0.03	0.70 ± 0.04	0.79 ± 0.06
PCLAF	$\mathbf{0.30 \pm 0.01}$	$\mathbf{0.29 \pm 0.01}$	0.74 ± 0.02	0.80 ± 0.01	0.83 ± 0.03	0.84 ± 0.04
RPCLAF	0.36 ± 0.00	0.34 ± 0.02	$\mathbf{0.80 \pm 0.03}$	$\mathbf{0.83 \pm 0.02}$	$\mathbf{0.85 \pm 0.05}$	$\mathbf{0.92 \pm 0.05}$
UCF	0.42 ± 0.01	0.38 ± 0.01	0.65 ± 0.01	0.75 ± 0.02	0.59 ± 0.01	0.73 ± 0.02
LCF	0.43 ± 0.01	0.37 ± 0.02	0.62 ± 0.01	0.74 ± 0.02	0.71 ± 0.01	0.83 ± 0.03
ACF	0.47 ± 0.01	0.58 ± 0.03	0.63 ± 0.02	0.72 ± 0.01	0.58 ± 0.01	0.70 ± 0.02
ULACF	0.47 ± 0.01	0.43 ± 0.01	0.73 ± 0.02	0.80 ± 0.02	0.76 ± 0.02	0.85 ± 0.05
SCF	0.39 ± 0.05	0.38 ± 0.02	0.70 ± 0.02	0.78 ± 0.04	0.67 ± 0.07	0.75 ± 0.06
POIC	0.49 ± 0.02	0.49 ± 0.02	0.71 ± 0.01	0.76 ± 0.03	0.65 ± 0.01	0.75 ± 0.02

Table 8
Impact of user numbers, in terms of RMSE.

	RMSE		
#(user)	60	90	119
CLAF	0.37 ± 0.02	0.36 ± 0.02	0.35 ± 0.02
PCLAF	$\mathbf{0.33 \pm 0.02}$	$\mathbf{0.31 \pm 0.02}$	$\mathbf{0.30 \pm 0.02}$
RPCLAF	0.38 ± 0.02	0.35 ± 0.02	0.35 ± 0.02
UCF	0.42 ± 0.02	0.42 ± 0.02	0.41 ± 0.02
LCF	0.43 ± 0.01	0.41 ± 0.01	0.41 ± 0.03
ACF	0.49 ± 0.01	0.47 ± 0.02	0.46 ± 0.01
ULACF	0.50 ± 0.04	0.49 ± 0.03	0.49 ± 0.01
SCF	0.40 ± 0.04	0.39 ± 0.06	0.37 ± 0.03
POIC	0.48 ± 0.02	0.48 ± 0.02	0.48 ± 0.02

our *RPCLAF*'s objective function is ranking-oriented, therefore its AUC performances on location and activity recommendations are shown to be the best through experiments. Our *PCLAF* and *RPCLAF* outperform *CLAF*, implying that personalization can be useful in recommendation. Besides, SCF can be seen as a special case of our *CLAF* algorithm, given that the auxiliary location and activity information is not used. Therefore, its performance is close to that of *CLAF*.

One interesting question to ask is whether we can simply make recommendation based on the POI counts, ignoring the user data. We see such an approach as a useful baseline, but we may not expect it to work as well as our algorithms due to several reasons. First, not using the user data makes us miss the chance of better understanding each single user's preferences for recommendation. For example, if we do not know a user likes going to do some gym after work, we may just recommend to her to enjoy some food rather than exercise around the area. Second, POI counts do not necessarily reflect the POI popularity. For example, we may see one place has only one or two restaurants, but they are both very nice, and thus attract a lot of customers to go there. If we do not consider the user data, we may not be able to discover such popularity information and use it for recommendation. Therefore, as we can see from Table 7, POIC baseline is quite competitive but still worse than our models.

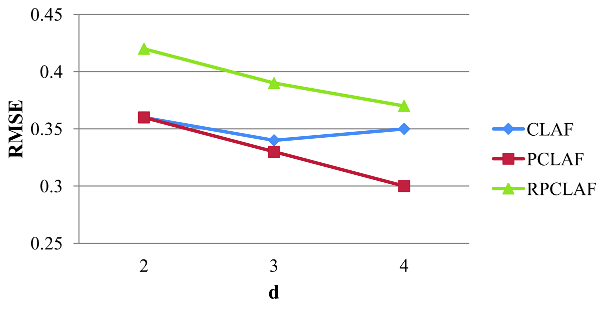
It is worth noting that in Table 7, the best RMSE value we achieved is 0.29 (using *PCLAF* with 50% of ratings as training data). As the rating data used in the experiments are normalized to be in the range of $[0, 1]$, such an RMSE value is not very good in fact. This shows that, the mobile recommendation problem is essentially a challenging problem: (i) the training data are usually limited (e.g. 50% of ratings only take up 1.7% of the tensor entries); (ii) the exact user rating pattern (i.e. how many times exactly a user performed some activity at some location) is not easy to predict. Because of these reasons, we develop the *RPCLAF* algorithm, which turns the exact rating prediction into preference prediction. In this way, our expected output is more consistent with the ranking nature of recommendation problem; and also, we can better utilize the limited amount of data by considering the additional pairwise preference.

6.4. Impact of user numbers

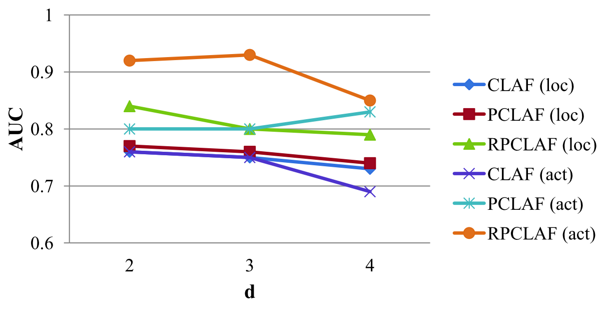
To evaluate the impact of the user number, we vary the number of users in building recommendation systems. Specifically, in this experiment, we randomly pick a fixed set of 60 users as testing users, and then change the number of training user from 60 to 119. We run the experiments five times, and report the results in terms of RMSE (see Table 8) and AUC (see Table 9). In general, as the number of user increases, the performances in terms of RMSE and AUC (both AUC_{loc} and AUC_{act}) increase. Besides, we also notice that, the performance improvement tends to diminish as training user numbers increases, implying that the performances for the specific set of test data tend to saturate.

Table 9
Impact of user numbers, in terms of AUC.

	AUC_{loc}			AUC_{act}		
#(user)	60	90	119	60	90	119
CLAF	0.70 ± 0.02	0.72 ± 0.01	0.72 ± 0.01	0.66 ± 0.07	0.69 ± 0.06	0.69 ± 0.02
PCLAF	0.72 ± 0.03	0.74 ± 0.02	0.74 ± 0.02	0.80 ± 0.06	0.82 ± 0.07	0.83 ± 0.04
RPCLAF	$\mathbf{0.75 \pm 0.05}$	$\mathbf{0.78 \pm 0.05}$	$\mathbf{0.78 \pm 0.03}$	$\mathbf{0.82 \pm 0.08}$	$\mathbf{0.84 \pm 0.06}$	$\mathbf{0.85 \pm 0.06}$
UCF	0.63 ± 0.03	0.63 ± 0.02	0.65 ± 0.03	0.56 ± 0.09	0.59 ± 0.02	0.60 ± 0.02
LCF	0.60 ± 0.02	0.64 ± 0.04	0.64 ± 0.02	0.70 ± 0.06	0.71 ± 0.05	0.72 ± 0.04
ACF	0.60 ± 0.04	0.64 ± 0.04	0.65 ± 0.03	0.58 ± 0.03	0.59 ± 0.02	0.60 ± 0.06
ULACF	0.68 ± 0.04	0.72 ± 0.03	0.71 ± 0.01	0.76 ± 0.02	0.79 ± 0.03	0.79 ± 0.05
SCF	0.68 ± 0.05	0.71 ± 0.03	0.71 ± 0.03	0.64 ± 0.06	0.67 ± 0.05	0.68 ± 0.03
POIC	0.69 ± 0.02	0.69 ± 0.02	0.69 ± 0.02	0.63 ± 0.04	0.63 ± 0.04	0.63 ± 0.04



(a) Impact of latent dimension (RMSE).



(b) Impact of latent dimension (AUC).

Fig. 11. Impact of latent factor dimension d.

Table 10
Impact of θ to *RPCLAF*.

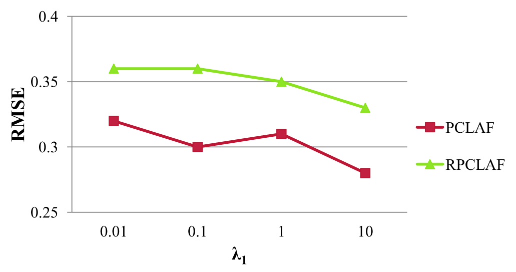
	RMSE	AUC_{loc}	AUC_{act}
$\theta = e^{0.1}$	0.36 ± 0.04	0.79 ± 0.02	0.87±0.06
$\theta = e^{1}$	0.37 ± 0.01	0.79 ± 0.03	0.85±0.08
$\theta = e^{2}$	0.35 ± 0.03	0.76 ± 0.02	0.82±0.07
$\theta = e^{3}$	0.35 ± 0.01	0.74 ± 0.02	0.80±0.05

6.5. *Impact of model parameters*

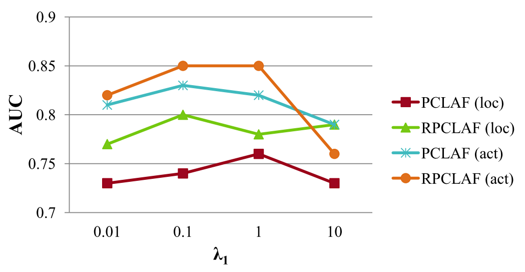
We also study the impact of the model parameters in our three algorithms, including λ_i $(i = 1, \ldots, 5)$ in *PCLAF* and *RPCLAF*, and β_j $(j = 1, 2, 3)$ in *CLAF*. In general, λ_1 controls the contribution of the user similarity input; λ_2 and β_1 controls the contribution of location features; λ_3 and β_2 control the contribution of activity correlations; λ_4 controls the contribution of user–location preferences; λ_5 and β_3 control the regularization. For each parameter, we vary its value from 0.01 to 10, and fix the other parameters (e.g. with value of 0.1). Then, we run the experiments five times, and report the average RMSE and AUC scores in Fig. 12. As shown in the figure, in general, the parameter values falling into $[0.1, 1]$ tend to give better performances, showing that reasonable weights are preferred on the these additional inputs for optimization. Besides, for activity correlations in Figs. 12(e) and 12(f), higher values for parameter λ_3 (and β_2) tend to give better results. This is possibly because, as opposed to other inputs, activity correlations have limited size (since the number of activities is much smaller than the number of users and the number of locations). Therefore, in order to encode this correlation constraint, we may need a higher weight in the objective function.

Similarly, in Fig. 11, we vary the latent factor dimension d from 2 to 4 (as the minimal dimension in the tensor is 5, i.e. the number of activities), and report the averaged RMSE and AUC scores. In general, under different model parameters, *RPCLAF* is better than *PCLAF* and *CLAF* in terms of AUC scores; in contrast, *PCLAF* is better than *CLAF* and *RPCLAF* in terms of RMSE.

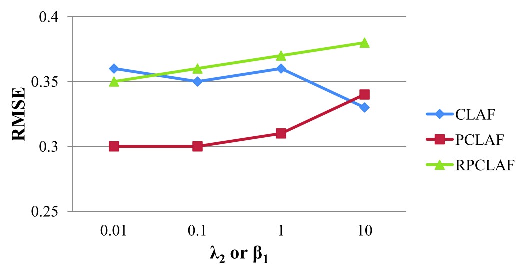
For *RPCLAF*, we also have a model parameter θ that controls the probability of rating ties in the logistic sigmoid function $\sigma(x) = \frac{1}{1+\theta e^{-x}}$. We study its impact to the performance of *RPCLAF*, and report the results in Table 10. From the table, we see that a bigger θ tends to pose a stronger constraint on modeling the rating ties, and too strong constraint may lead to performance drop.



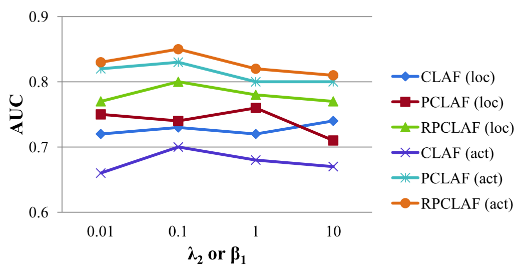
(a) Impact of user similarity (RMSE).



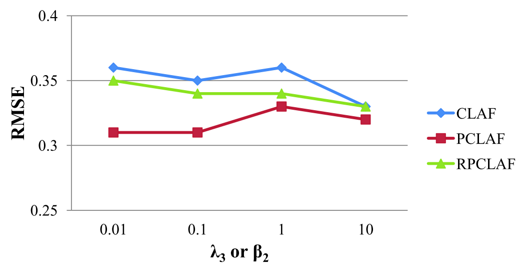
(b) Impact of user similarity (AUC).



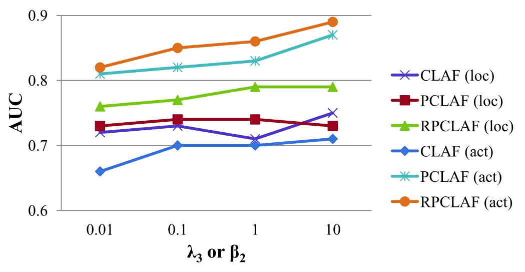
(c) Impact of location features (RMSE).



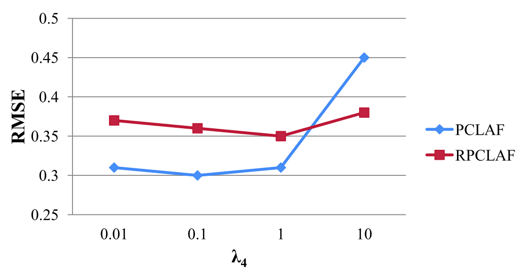
(d) Impact of location features (AUC).



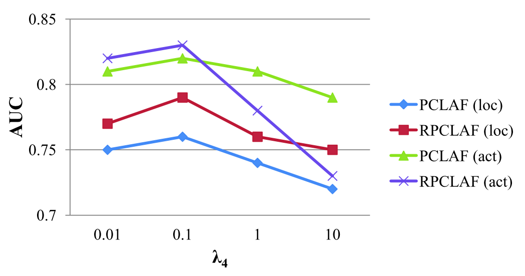
(e) Impact of activity correlation (RMSE).



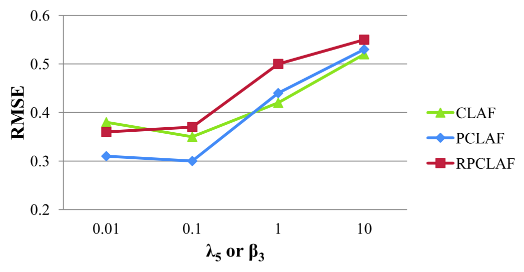
(f) Impact of activity correlation (AUC).



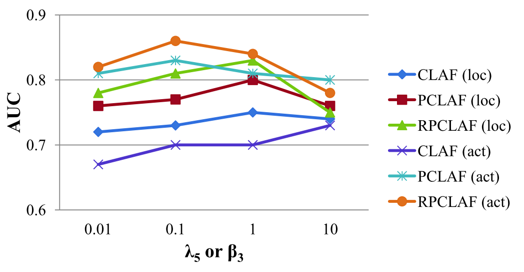
(g) Impact of user-location input (RMSE).



(h) Impact of user-location input (AUC).



(i) Impact of regularization (RMSE).



(j) Impact of regularization (AUC).

Fig. 12. Impact of model parameters, where "CLAF (loc)" (or, "CLAF (act)") in the plots indicates the AUC_{loc} (or, AUC_{act}) score for *CLAF* algorithm.

Table 11
Impact of AUC tolerance parameter ϵ, with 30% of data used for training.

	AUC_{loc}			AUC_{act}		
ϵ	0.05	0.1	0.2	0.05	0.1	0.2
CLAF	0.68 ± 0.02	0.73 ± 0.01	0.79 ± 0.03	0.60 ± 0.04	0.67 ± 0.02	0.79 ± 0.07
PCLAF	0.70 ± 0.01	0.74 ± 0.02	0.81 ± 0.02	0.68 ± 0.03	0.83 ± 0.03	0.92 ± 0.04
RPCLAF	$\mathbf{0.77 \pm 0.03}$	$\mathbf{0.79 \pm 0.03}$	$\mathbf{0.85 \pm 0.02}$	$\mathbf{0.70 \pm 0.05}$	$\mathbf{0.85 \pm 0.08}$	$\mathbf{0.95 \pm 0.04}$
UCF	0.63 ± 0.04	0.65 ± 0.03	0.65 ± 0.01	0.59 ± 0.01	0.60 ± 0.04	0.62 ± 0.02
LCF	0.61 ± 0.03	0.62 ± 0.04	0.65 ± 0.04	0.68 ± 0.03	0.71 ± 0.05	0.73 ± 0.04
ACF	0.58 ± 0.02	0.58 ± 0.05	0.64 ± 0.02	0.57 ± 0.01	0.63 ± 0.03	0.64 ± 0.03
ULACF	0.70 ± 0.02	0.73 ± 0.02	0.75 ± 0.02	0.73 ± 0.01	0.79 ± 0.02	0.81 ± 0.01
SCF	0.67 ± 0.01	0.71 ± 0.01	0.77 ± 0.03	0.62 ± 0.05	0.68 ± 0.01	0.76 ± 0.06
POIC	0.66 ± 0.03	0.71 ± 0.01	0.71 ± 0.01	0.59 ± 0.02	0.65 ± 0.01	0.71 ± 0.01

Table 12
Comparison with trivial recommender, with different percentages of data used for training.

	RMSE		AUC_{loc}		AUC_{act}	
Percent	30%	50%	30%	50%	30%	50%
CLAF	0.35 ± 0.02	0.36 ± 0.03	0.79 ± 0.03	0.86 ± 0.02	0.79 ± 0.07	0.90 ± 0.06
PCLAF	0.30 ± 0.01	0.29 ± 0.01	0.81 ± 0.02	0.88 ± 0.01	0.92 ± 0.04	0.96 ± 0.03
RPCLAF	0.36 ± 0.00	0.34 ± 0.02	$\mathbf{0.85 \pm 0.02}$	$\mathbf{0.90 \pm 0.03}$	$\mathbf{0.96 \pm 0.03}$	$\mathbf{0.98 \pm 0.03}$
Trivial	$\mathbf{0.27 \pm 0.00}$	$\mathbf{0.26 \pm 0.01}$	0.75 ± 0.01	0.79 ± 0.00	0.94 ± 0.02	0.95 ± 0.00

6.6. Impact of AUC parameter

In our AUC score, we have a tolerance parameter ϵ to measure the prediction ties, so that two predictions having a difference less than ϵ are seen to be a tied order in ranking. We study its impact in evaluating all the algorithms. As shown in Table 11, as ϵ increases, the AUC scores tend to be higher. That is because, with higher tolerance, the accuracy of predicting the tied order becomes higher. Besides, it is also shown that our algorithms, especially *RPCLAF*, can consistently outperform the baselines.

6.7. Comparison with a trivial recommender

We also compare our three algorithms with a trivial recommender, which always uses the average non-zero rating values of training data as the prediction. This baseline is interesting w.r.t. our data characteristics. Our data have many non-zero rating values as "1", with the global mean rating value of 1.43 and standard deviation of 0.70 (before rating values are normalized to $[0, 1]$). Such a data property may benefit the trivial recommender. In the experiments, we set $\lambda_1 = \lambda_2 = \lambda_4 = \lambda_5 = 0.1$, $\lambda_3 = 1$, $\beta_1 = \beta_3 = 0.1$, $\beta_2 = 1$, $k = 4$, $\theta = e^1$, $\epsilon = 0.2$. As shown in Table 12, in most of the time, our proposed algorithms can outperform the trivial recommender in terms of AUC scores. Our algorithms work better than the trivial recommender especially in location recommendation. This is because the rating variance is generally bigger, given that the location number is bigger than the activity number. The trivial recommender seems to work well in terms of RMSE, by benefiting from the relatively small standard deviation in the rating values. But the best RMSE results our algorithms can achieve are comparable. Finally, we note that though the trivial recommender seems to work well by benefiting from our dataset's property (i.e. with many ratings of "1"), we expect our algorithm to generalize well to other datasets that are not necessarily biased to some rating values.

6.8. Discussion

It is worth noting that, like most of the existing collaborative filtering work [10,9,17,21], our proposed algorithms do not make any specific assumption on how the missing data are generated. It is generally believed in the collaborative filtering literatures that the values are missing at random. In other words, a rating that is missing does not depend on the value of that rating, or the value of any other missing ratings. However, some recent research such as [24] points out that values are not necessarily missing at random. Consider our problem: suppose that a user's preference of doing some activity a at a location l is low, then we are unlikely to have collected data on this pattern. As a result, the missing data in our sample are biased towards "low-rated" user–location–activity entries. This may possibly skew the hold-out evaluation. Marlin and Zemel proposed to formulate the missing data generation with a probabilistic mixture model to address this problem [24]. In the future, we are interested in extending our algorithms along this line.

Another interesting problem, yet to be studied more in the future, is that our way to generate the ratings is different from traditional rating system. Recall that we define a user–location–activity rating as the mention count value in Eq. (3). Because the users may have omitted some instances of a particular activity at a particular location from their comments, the ratings we get in our sample could be lower than the "true" value. This can also lead to some missing data. In the future, we are also interested in further exploiting this issue.

7. Related work

In the past, little work studying collaborative location and activity recommendations has been done. Most of the previous work focused on either recommending some specific types of locations [25–28], or only recognizing the user activities from sensor data rather than providing location and activity recommendations together [29,30].

7.1. Location recommendation

Location recommendation has been an important topic in geo-related services. Some systems, based on an individual user's current location, retrieve important surrounding locations and their contexts for recommendations. For example, in [31], a mobile application framework, which enables a mobile phone user to query the geo-coded Wikipedia articles for landmarks in the vicinity, is presented. In [32], a Cyberguide system is developed to provide the librarian information which describes the nearby buildings and related people identities. Comparatively, our system exploits the user location histories and recommends the interesting locations all round the city instead of only nearby locations.

There are some systems focusing on recommending some specific types of locations. For example, in [25], a CityVoyager system is developed to recommend shops. It collects the users' shopping histories based on GPS logs, and uses an item-based collaborative filtering method to recommend to a user some shops that are similar to his/her previously visited shops. In [27], a system considering both users' preferences and location contexts is shown to recommend restaurants. It uses Bayesian learning to calculate some recommendation values for restaurants so as to provide a ranking list for recommendation. Similarly, in [26], a Geowhiz system, which uses a user-based collaborative filtering algorithm to recommend restaurants, is proposed. In [33], the recommended locations are hot spots for tourism. A HITS-based model is proposed to take into account a user's travel experience and the interest of a location in recommendation, so that only the locations that are really popular and also recommended by experienced users can be recommended. In contrast to those systems limited to modeling only one type of location for recommendations, our system is capable of handling various types of locations. That is, we can recommend locations not only for food and drinks but also for shopping, and so on.

7.2. Activity recommendation

Activity recommendation is a pretty new research issue with little research done on it so far [34]. Yet it is a quite common question in our daily life to ask what we can do if we want to visit some place. Most of the previous work related to the study focuses on how to recognize an activity from various sensor data such as GPS [35], RFID [36], motion sensor [37] or WiFi [38] by ubiquitous computing [16].

Early activity recognition algorithms are based on logic and usually described as a logical inference process w.r.t. a set of first-order statements [39]. However, with the development of the sensor technology, these logic-based approaches were found generally limited in modeling uncertainty and noise of the sensor data. As a result, the learning-based algorithms were introduced to model the relationships between the sensor observations and the activities in a sophisticated way by machine learning. For example, in [29], the Hidden Markov Model is used to model the sequential object sensor observations for fine-grained activity recognition. While in [30], a supervised decision tree is proposed to recognize ADLs. Notice that, most of these studies only consider fine-grained activity recognition in indoor environments, and they did not consider using user location histories to model a user's activities in outdoor environments. In this paper, we show how to parse user GPS location data, and use them together with the mined location features and activity correlations to provide both indoor and outdoor coarse-grained activity recommendations w.r.t. location queries.

Some other work related to outdoor activity recognition includes [35,40,41]. For example, in [35], based on GPS data, a supervised hierarchical conditional random field model is used to recognize whether a user is at work, sleeping at home, or visiting friends, and so on. Both the studies in [40] and [41] are based on a reality mining project in MIT, which uses mobile phones as the sensors for recording the user's movements and social behaviors. Unsupervised learning algorithms, such as Principle Component Analysis (PCA), Latent Dirichlet Allocation (LDA) and Author Topic model (ATM), are applied to the user's location data to discover the frequent patterns of user's activities. Compared with these studies, our work not only predicts what kind of activities are suitable for some location, but also well integrates it with the location recommendations.

8. Conclusion

In this paper, we studied how to use real-world GPS data to retrieve relevant mobile information for answering two typical questions. The first question is, if we want to do something, where shall we go? This question corresponds to location recommendation. The second question is, if we visit some place, what can we do there? This question corresponds to activity recommendation. We show that these two questions are inherently related, as they can be seen as a collaborative filtering problem in a user–location–activity rating tensor. We propose three algorithms to solve this problem. The first one, *CLAF*, is a matrix-based CF model which aims to minimize the square loss of missing entry value predictions in the location–activity matrix (without modeling user) [3]. The second one, *PCLAF*, is a tensor-based CF model which aims to minimize the square loss of missing entry value predictions in the user–location–activity tensor [4]. Compared with *CLAF*, *PLCAF* takes user into account for optimization and thus is able to provide personalized recommendation. The third one, *RPCLAF*, is a tensor-based CF model which aims to minimize the ranking loss on missing entries in the user–location–activity tensor. Compared with *CLAF* and *PCLAF*, this newly proposed *RPCLAF* model considers recommendation as a ranking problem and thus focused on directly optimizing the ranking performance.

Because the user–location–activity tensor is very sparse in practice, we also propose to exploit other information, including user–user similarities, location features, activity–activity correlations and user–location visiting preferences from various information sources, to enhance the performance. We extensively evaluated our system on a real-world GPS dataset. We show that, our three algorithms can consistently outperform six competing baselines. Particularly, on average,[7] our newly proposed *RPCLAF* algorithm can achieve at least 7% improvement on location recommendation (in terms of AUC score) and 10% improvement on activity recommendation, compared with the best performances of all these six baselines. Besides, on average, our *RPCLAF* algorithm also achieves at least 6% improvements on location recommendations and 6% improvements on activity recommendations, compared with the best performances of our two previous algorithms *CLAF* and *PCLAF*.

In the future, we will consider more external information, such as incorporating the time or sequence information of the trajectories to provide more constraints in the recommendations. Besides, we are also interested in studying how to update our models in an online fashion as more users accumulate data continuously. Meanwhile, we are also interested in integrating our models with cloud computing platforms so as to handle the large number of users.

Acknowledgements

We thank Hong Kong RGC project 621010 for supporting the research. We also thank the anonymous reviewers for their helpful comments and constructive suggestions.

References

[1] P. Symeonidis, A. Nanopoulos, Y. Manolopoulos, Tag recommendations based on tensor dimensionality reduction, in: Proc. of the ACM Conference on Recommender Systems, 2008, pp. 43–50.

[2] A. Cichocki, R. Zdunek, A.H. Phan, S.-i. Amari, Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multiway Data Analysis and Blind Source Separation, Wiley, 2009.

[3] V.W. Zheng, Y. Zheng, X. Xie, Q. Yang, Collaborative location and activity recommendations with gps history data, in: Proc. of the 19th International World Wide Web Conference (WWW '10), ACM, New York, NY, USA, 2010.

[4] V.W. Zheng, B. Cao, Y. Zheng, X. Xie, Q. Yang, Collaborative filtering meets mobile recommendation: A user-centered approach, in: Proc. of the 24th AAAI Conference on Artificial Intelligence (AAAI'10), Atlanta, Georgia, USA, 2010, pp. 236–241.

[5] Y.-F. Chen, G. Di Fabbrizio, D. Gibbon, R. Jana, S. Jora, B. Renger, B. Wei, GeoTV: navigating geocoded RSS to create an IPTV experience, in: Proc. of the 16th International Conference on World Wide Web (WWW '07), 2007.

[6] L. Liao, D. Fox, H.A. Kautz, Learning and inferring transportation routines, Artificial Intelligence (2007) 311–331.

[7] Y. Zheng, X. Zhou (Eds.), Computing with Spatial Trajectories, Springer, 2011.

[8] Y. Zheng, X. Xie, W.-Y. Ma, GeoLife, A collaborative social networking service among user, location and trajectory, IEEE Database Eng. Bull. (2010).

[9] J. Wang, A.P. de Vries, M.J.T. Reinders, Unifying user-based and item-based collaborative filtering approaches by similarity fusion, in: Proc. of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06), 2006, pp. 501–508.

[10] N. Srebro, T. Jaakkola, Weighted low-rank approximations, in: Proc. of the 21st International Conference on Machine Learning (ICML '03), 2003, pp. 720–727.

[11] Y. Zheng, L. Liu, L. Wang, X. Xie, Learning transportation mode from raw gps data for geographic applications on the web, in: Proc. of the 17th International Conference on World Wide Web (WWW '08), 2008, pp. 247–256.

[12] Q. Li, Y. Zheng, X. Xie, Y. Chen, W. Liu, W.-Y. Ma, Mining user similarity based on location history, in: Proc. of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '08), 2008, pp. 1–10.

[13] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, Optics ordering points to identify the clustering structure, SIGMOD Rec. 28 (2) (1999) 49–60.

[14] K. Nigam, A.K. McCallum, S. Thrun, T. Mitchell, Text classification from labeled and unlabeled documents using EM, Mach. Learn. 39 (2000) 103–134.

[15] C.D. Manning, P. Raghavan, H. Schutze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[16] V.W. Zheng, D.H. Hu, Q. Yang, Cross-domain activity recognition, in: Proc. of the 11th International Conference on Ubiquitous Computing (UbiComp '09), 2009, pp. 61–70.

[17] A.P. Singh, G.J. Gordon, Relational learning via collective matrix factorization, in: Proc. of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08), 2008, pp. 650–658.

[18] G. Adomavicius, R. Sankaranarayanan, S. Sen, A. Tuzhilin, Incorporating contextual information in recommender systems using a multidimensional approach, ACM Trans. Inf. Syst. 23 (2005) 103–145.

[19] N.N. Liu, M. Zhao, Q. Yang, Probabilistic latent preference analysis for collaborative filtering, in: Proc. of the 18th ACM Conference on Information and Knowledge Management, CIKM '09, ACM, New York, NY, USA, 2009, pp. 759–766.

[20] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proc. of the 8th IEEE International Conference on Data Mining, ICDM '08, IEEE Computer Society, Washington, DC, USA, 2008, pp. 263–272.

[21] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proc. of the 25th Conference on Uncertainty in Artificial Intelligence, UAI '09, 2009.

[22] K. Zhou, G.-R. Xue, H. Zha, Y. Yu, Learning to rank with ties, in: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '08, ACM, New York, NY, USA, 2008, pp. 275–282.

[7] Under different percentages of training data used.

[23] J.L. Herlocker, J.A. Konstan, A. Borchers, J. Riedl, An algorithmic framework for performing collaborative filtering, in: Proc. of the 22nd Annual ACM SIGIR Conference, SIGIR '99, ACM, New York, NY, USA, 1999, pp. 230–237.
[24] B.M. Marlin, R.S. Zemel, Collaborative prediction and ranking with non-random missing data, in: Proceedings of the Third ACM Conference on Recommender Systems, RecSys '09, ACM, New York, NY, USA, 2009, pp. 5–12.
[25] Y. Takeuchi, M. Sugimoto, CityVoyager: An outdoor recommendation system based on user location history, in: Proc. of Ubiquitous Intelligence and Computing, 2006, pp. 625–636.
[26] T. Horozov, N. Narasimhan, V. Vasudevan, Using location for personalized POI recommendations in mobile environments, in: Proc. of the International Symposium on Applications on Internet, 2006, pp. 124–129.
[27] M.-H. Park, J.-H. Hong, S.-B. Cho, Location-based recommendation system using Bayesian user's preference model in mobile devices, in: Proc. of Ubiquitous Intelligence and Computing, 2007, pp. 1130–1139.
[28] H. Yoon, Y. Zheng, X. Xie, W. Woo, Smart itinerary recommendation based on user-generated GPS trajectories, in: Proc. of Ubiquitous Intelligence and Computing (UIC '10), 2010.
[29] D.J. Patterson, D. Fox, H.A. Kautz, M. Philipose, Fine-grained activity recognition by aggregating abstract object usage, in: Proc. of the 9th IEEE International Symposium on Wearable Computers (ISWC '05), IEEE Computer Society, Washington, DC, USA, 2005, pp. 44–51.
[30] M.R. Hodges, M.E. Pollack, An 'object-use fingerprint': The use of electronic sensors for human identification, in: Proc. of the 9th International Conference on Ubiquitous Computing (UbiComp '07), 2007, pp. 289–303.
[31] R. Simon, P. Frölich, A mobile application framework for the geospatial web, in: Proc. of the 16th International Conference on World Wide Web (WWW '07), 2007.
[32] G. Abowd, C. Atkeson, J. Hong, S. Long, R. Kooper, M. Pinkerton, Cyberguide: a mobile context-aware tour guide, Wirel. Netw. (1997) 421–433.
[33] Y. Zheng, L. Zhang, X. Xie, W.-Y. Ma, Mining interesting locations and travel sequences from GPS trajectories, in: Proc. of the 18th International Conference on World Wide Web (WWW '09), 2009, pp. 791–800.
[34] V. Bellotti, B. Begole, E.H. Chi, N. Ducheneaut, J. Fang, E. Isaacs, T. King, M.W. Newman, K. Partridge, B. Price, P. Rasmussen, M. Roberts, D.J. Schiano, A. Walendowski, Activity-based serendipitous recommendations with the Magitti mobile leisure guide, in: Proc. of the 26th Annual SIGCHI Conference on Human Factors in Computing Systems, CHI '08, ACM, New York, NY, USA, 2008, pp. 1157–1166.
[35] L. Liao, D. Fox, H.A. Kautz, Location-based activity recognition, in: Proc. of Advances in Neural Information Processing Systems (NIPS '05), 2005.
[36] D. Wyatt, M. Philipose, T. Choudhury, Unsupervised activity recognition using automatically mined common sense, in: Proc. of the Twentieth National Conference on Artificial Intelligence (AAAI '05), 2005, pp. 21–27.
[37] S.S. Intille, K. Larson, E.M. Tapia, J. Beaudin, P. Kaushik, J. Nawyn, R. Rockinson, Using a live-in laboratory for ubiquitous computing research, in: Proc. of the 4th International Conference on Pervasive Computing (Pervasive '06), 2006, pp. 349–365.
[38] J. Yin, Q. Yang, J.J. Pan, Sensor-based abnormal human-activity detection, IEEE Trans. Knowl. Data Eng. 20 (8) (2007) 17–31.
[39] H. Kautz, A formal theory of plan recognition, Ph.D. thesis, University of Rochester, 1987.
[40] K. Farrahi, D. Gatica-Perez, What did you do today?: Discovering daily routines from large-scale mobile data, in: Proc. of the 16th ACM International Conference on Multimedia (ACM MM '08), 2008, pp. 849–852.
[41] N. Eagle, A. Pentland, Eigenbehaviors: Identifying structure in routine, Behav. Ecol. Sociobiol. (2009) 1057–1066.