Collaborative Location and Activity Recommendations with GPS History Data

Vincent W. Zheng[†], Yu Zheng[‡], Xing Xie[‡], Qiang Yang[†]
[†] Hong Kong University of Science and Technology
[‡] Microsoft Research Asia, 4F, Sigma Building, No.49 Zhichun Road, Haidian District, Beijing 100190, China
[†] {vincentz, qyang}@cse.ust.hk, [‡] {yuzheng, xingx}@microsoft.com

ABSTRACT

With the increasing popularity of location-based services, such as tour guide and location-based social network, we now have accumulated many location data on the Web. In this paper, we show that, by using the location data based on GPS and users' comments at various locations, we can discover interesting locations and possible activities that can be performed there for recommendations. Our research is highlighted in the following location-related queries in our daily life: 1) if we want to do something such as sightseeing or food-hunting in a large city such as Beijing, where should we go? 2) If we have already visited some places such as the Bird's Nest building in Beijing's Olympic park, what else can we do there? By using our system, for the first question, we can recommend her to visit a list of interesting locations such as Tiananmen Square, Bird's Nest, etc. For the second question, if the user visits Bird's Nest, we can recommend her to not only do sightseeing but also to experience its outdoor exercise facilities or try some nice food nearby. To achieve this goal, we first model the users' location and activity histories that we take as input. We then mine knowledge, such as the location features and activity-activity correlations from the geographical databases and the Web, to gather additional inputs. Finally, we apply a collective matrix factorization method to mine interesting locations and activities, and use them to recommend to the users where they can visit if they want to perform some specific activities and what they can do if they visit some specific places. We empirically evaluated our system using a large GPS dataset collected by 162 users over a period of 2.5 years in the real-world. We extensively evaluated our system and showed that our system can outperform several state-of-the-art baselines.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *data mining*. H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *clustering, information filtering*. H.5.2 [**Information Interface and Presentation**]: User Interface.

General Terms

Algorithms, Design, Experimentation

Keywords

Location and Activity Recommendations, Collaborative Filtering

1. INTRODUCTION

As the mobile devices with positioning function, such as GPS-phones, become more and more popular, people now are able to know their locations easily. Based on these location data, various location-based services are provided on the Web and shown to be quite attractive to the users. For example, a bunch of outdoor sports forums have emerged to provide various geo-related Web services [1][2][20]. By using these services, the forum users can upload and share their outdoor sports (such as bicycle riding) trajectories with other users. In this way, they can conveniently manage their own outdoor sports trajectories and also share them with other outdoor sports fans. In addition, thanks to some Web-based location data management services [7], the users can now share on Web not only their raw GPS trajectories with coordinates and time stamps, but also comments denoting what the user did, what she saw and/or how she felt on some locations. Figure 1 gives an example of such a GPS data management system: a user uploaded a GPS trajectory to Forbidden City area in Beijing, and he also attached some comments (depicted as small pink boxes, each unfolded as a text box) about how he felt about the places. Such comments bring more semantics to the GPS trajectories, and make it easier for GPS users to share their travel experiences. Beyond directly sharing the GPS trajectories, we can also better understand the location trajectories by mining knowledge from the users' location trajectories. In this way, we are capable to provide more interesting location-based services, including transportation routine prediction [3][4], location-based activity recognition [5] and location-based social network [6,22].

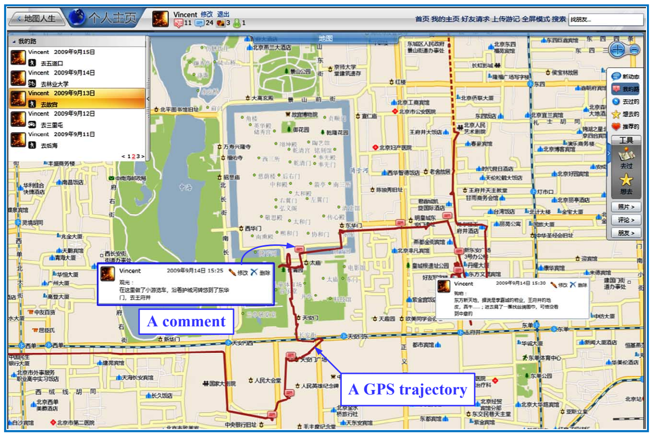


Figure 1. GPS data management services

In this paper, we aim to mine more knowledge from the GPS location data, so that we can answer two typical questions that we often ask in our daily: 1) if we want to do something such as sightseeing or food-hunting in a large city such as Beijing, where should we go? 2) If we have already visited some places such as the Bird's Nest building in Beijing's Olympic park, what else can we do there? In general, the first question corresponds to *location recommendation* given some activity query (where "activity" can refer to various human behaviors such as food-hunting, shopping, watching movies/shows, enjoying sports/exercises, tourism, etc.), and the second question corresponds to *activity recommendation* given some location query. By answering these two typical questions, we can satisfy many information needs for the users in both their daily routines and trip planning. We show to put both location recommendation and activity recommendation together in our knowledge mining, since locations and activities are closely related in nature. Specifically, to model the relationship between the locations and the activities, we can construct a location-activity matrix (details are given in Section 3.2), whose rows denote the locations and columns denote the activities. Each entry in this location-activity matrix is a rating showing how often an activity is performed in a location. Therefore, as shown in Figure 2, we can see location recommendation given some activity query as ranking over the rows given some column, and activity recommendation as ranking over the columns given some row.

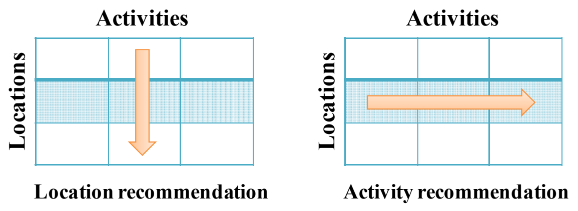


Figure 2. Illustration for location & activity recommendations

However, it is not easy to obtain such a complete location-activity matrix for location and activity recommendations from the raw GPS data due to the following reasons: 1) the ratings in such a location-activity matrix are not easy to get from the raw GPS data with merely location coordinates and timestamps. Recall that a rating in the matrix denotes how often an activity is performed in a location, so we may need to know what each user did on that location to get a rating. But the raw GPS data may not convey such information, and we have to find another way. We propose to use the possibly available comments provided by the users, which indicate what the user did on some locations, as shown in Figure 1. But unfortunately, in practice the users usually do not provide many comments. For example, in our dataset which is collected from a Web-based GPS data management service for over 2.5 years, we have 12,765 GPS trajectories, but only 530 comments and many of them were attached to some same popular locations. It means, many locations do not have any comments attached to them, so when we try to get the ratings from these comments, we may have many missing entries in the matrix. 2) Based on the previous reason, we can only get a very sparse location-activity matrix (e.g. in our dataset, we have less than 0.6% entries with non-missing values), so it is difficult to do recommendations with such limited information. We suggest exploiting other additional information on the locations and activities, and use it to alleviate the data sparsity. However, what kind of information we should extract? How we can incorporate it with the location-activity matrix to do recommendations? These are non-trivial questions.

In this paper, based on the GPS history data, including location information (i.e. coordinates and timestamps) and some available user comments, we develop a system to provide both location and activity recommendations. We achieve this goal by exploiting various useful information sources, i.e. meaningful location features and activity-activity correlations, and using them by collaborative filtering with the sparse location-activity matrix to do recommendations. Our collaborative location and activity recommendation (CLAR) model is based on collective matrix factorization to propagate information among the two additional information sources and the sparse location-activity matrix, so that we can collaboratively predict the missing entries in the location-activity matrix for recommendations. Our work is a step towards associating the locations and the activities to boost the location-based services on the Web by using mobile data. The contributions of this paper lie in three aspects:

- We put forward a new problem for collaborative location and activity recommendations based on the GPS history data, so that we can provide more specific recommendations with location or activity constraints.
- We propose to exploit location features and activity-activity correlations for collaborative filtering, so as to address the data sparsity problem of the GPS histories. We also show how to well incorporate this additional information with the incomplete location-activity matrix in a collective matrix factorization model for final recommendations.
- We evaluate our system using a large GPS dataset, which was collected by 162 users over a period of 2.5 year in the real world. The number of GPS points is around 4 million and its total distance was over 139,310 kilometers.

The remainder of this paper is organized as follows. Section 2 gives an overview of our system. Section 3 introduces the data modeling for location-activity matrix generation, location feature and activity-activity correlation extraction. Section 4 details our collaborative filtering model which takes the previous three pieces of information as inputs. In Section 5, we report the experimental results and offer some discussions. In Section 6, we survey the related works. In Section 7, we draw our conclusions and present the future work.

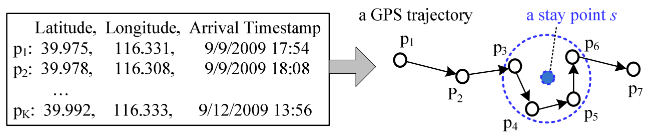
2. OVERVIEW OF OUR SYSTEM

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

2.1 Preliminary

First, we will 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: $Traj = \langle p_0, p_1, \dots, p_k \rangle$, where a GPS point $p_i = (x_i, y_i, t_i)$, $\forall 0 \le i < k$, with t_i as a timestamp ($t_i < t_{i+1}$), and (x_i, y_i) as the two-dimension coordinates [4]. In the right part of Figure 3, we show a trajectory consisted of 7 GPS points.



	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

Figure 3. GPS trajectory and stay point

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 $P = \langle p_m, p_{m+1}, \dots, p_n \rangle$, where $\forall m < i \le n$, $Dist(p_m, p_i) \le D_r$, $Dist(p_m, p_{n+1}) > D_r$ and $Int(p_m, p_n) \ge T_r$. Hence, a stay point $s = (x, y, t_a, t_l)$, where

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

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 [4].

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 [4][6]. However, in practice, when we consider many GPS trajectories together, we may find that some stay points refer to a 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 recommendation, we focus on a whole interested region such as Bird's Nest rather than its two wings, so we need to further extract some geographical region by clustering the nearby stay points. We call these regions as stay regions.

Definition 3. Stay region (location): Given all the stay points extracted from the GPS data as $S = \{s_1, s_2, \dots, s_N\}$ 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 $S' = \{s'_m, s'_{m+1}, \dots, s'_n | s'_i \in S, \forall m \le i \le n\}$ belonging to some same cluster. Hence, a stay region $r = (x, y)$, where

$$r.x = \sum_{i=m}^{n} s_i.x/|S'|, \; r.y = \sum_{i=m}^{n} s_i.y/|S'|, \quad (2)$$

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 as shown in Figure 6. Notice that we do not directly extract stay regions by clustering on the raw GPS points from all the trajectories. This is because we may lose the sequential information by mixing the raw GPS points from different trajectories together, and thus it is hard to detect the meaningful stays.

2.2 Application Scenarios

The work reported in this paper is an important component of our GeoLife project [7], whose prototype has been internally accessible within Microsoft since Oct. 2007. So far, we have had 162 individuals using this system.

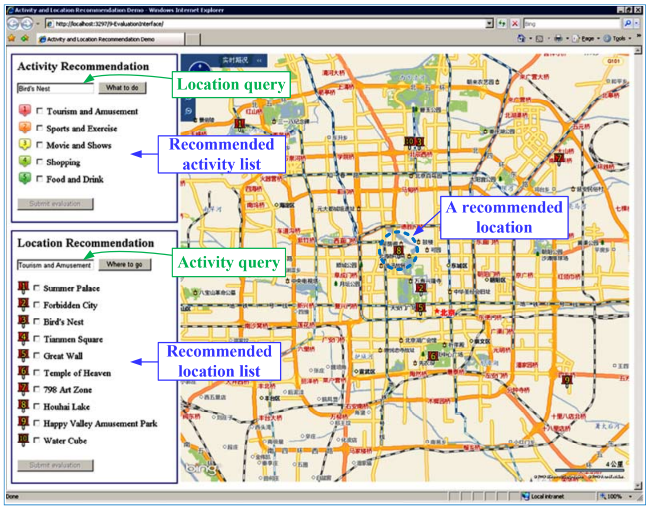
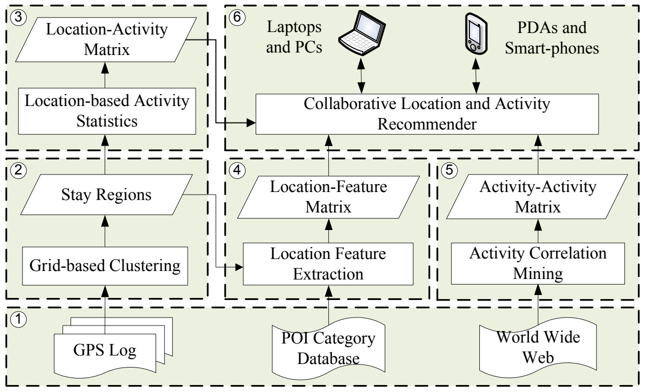


Figure 4. User interface for our system

Figure 4 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, for example, in activity recommendation, a 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 5 here). The user can provide some feedbacks 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 10 here) and display them on the map, so that the user can zoom in on the map and get more details (e.g. transportations). The user can also view the location candidates ranked lower than 10 to get more recommendations. Similarly, the user can also provide feedbacks on location recommendation.

2.3 Architecture

We demonstrate our system's architecture in Figure 5. Our system consists of 6 parts, including data inputs, stay region extraction, location-activity information extraction, location feature extraction, activity-activity correlation mining and collaborative location and activity recommendations. In the first 5 parts, we model the data and extract knowledge as inputs to train a recommendation system. This process can be performed off-line. In real-time (for part 6), the users can access the recommender through internet using laptops/PCs or PDAs/smart-phones, and submit the query (i.e. activity or location names). Our system will then return a ranking list of locations or activities given the activity or location query.



① Data Inputs ② Stay Region Extration ③ Location-Activity Extraction ④ Location-Feature Extraction ⑤ Activity Correlation Mining ⑥ Collaborative Loc. & Act. Recommendations

Figure 5. Architecture of our system

Data inputs: In addition to the users' GPS trajectories with some comments, our system also exploits various information sources, including Point-of-Interest (POI) category database and World Wide Web, to alleviate the data sparsity problem that occurs when there are few comments to get reliable statistics of the location-activity relations. We will give more details about using these two information sources in Section 3.2 and Section 3.3.

Stay region extraction: As the stay points sometimes may refer to some common locations, we extract stay regions by clustering the stay points and use them for location recommendations. Notice that in practice, the recommended locations are supposed to have limited region sizes, so we take this constraint into consideration and propose a grid-based clustering algorithm to extract the stay regions. More details are in Section 3.1.

Location-activity information extraction: With the available user comments to the GPS trajectories, we can get the statistics about what kinds of activities the users performed on some location, and how often they performed these activities. By organizing this statistics' data in a matrix form, we can have a location-activity matrix, with rows as locations and columns as activities. An entry in the matrix denotes the frequency for the users to perform some activity on some location. We will give the details in Section 3.2 to show how to get these entries. Note that, due to the limited amount of comments, the obtained location-activity matrix is quite sparse. Our ultimate objective is to appropriately fill all the missing entries in that matrix, so that we can rank all the entries for collaborative location and activity recommendation.

Location feature extraction: We exploit the location features with the help of POI category database. The database is based on the city yellow pages, and it can provide us the knowledge that what kinds of POIs we have in an area. For example, by query the POI category database with some location area, we can know how many restaurants (and theaters, museums, etc.) exist in this area. This helps us to get some sense of this location's functionalities, so that we can use them as features for better recommendations. Similarly, by organizing the data in a matrix form, we can have a location-feature matrix, with rows as locations and columns as features. Each entry of the matrix denotes some feature value on that location. We give more details in Section 3.3.

Activity-activity correlation mining: We exploit the World Wide Web, to get the knowledge about the activity correlations. With this knowledge, we may better infer that if a user performs some activity on a location, then it is likely that she will also perform another activity. For example, when the users go to see a movie in some place, it is quite likely they will also have some foods/drinks there. One possible way to get such activity-activity correlation information is directly having some statistics over the activity occurrence in the GPS data; however, as the amount of available comments is few, we may not get reliable statistics. Therefore, we refer to the World Wide Web, which is a huge knowledge source, to get such statistics. By organizing the data in a matrix form, we have an activity-activity matrix, with rows and columns both as activities. Each entry of the matrix denotes the correlation between a pair of activities. We give more details in Section 3.4.

Collaborative location and activity recommendations: Having the knowledge of location-activity matrix, location-feature matrix and activity-activity correlation matrix, we can train a recommender system. We propose a collaborative filtering model under the collective matrix factorization framework [11], and manage to fill the missing entries in the location-activity matrix. Based on the filled location-activity matrix, we will rank and retrieve the top k locations/activities for recommendations to the users who access our system by PCs/PDAs. More details are given in Section 4.

3. DATA MODELING

In this section, we will introduce how to model the data in order to obtain the location-activity, location-feature and activity-activity matrices as inputs for training the recommender.

3.1 Stay Region Extraction

In practice, the recommended locations should not be too large in size; otherwise, the user may not easily find the true interesting locations in a large area. As a result, when we consider clustering the stay points to get stay regions, we need to take such a limit into account. Previous clustering algorithms used in GPS data processing, such as the classic k-means algorithm and the density-based OPTICS clustering algorithm [19], do not constrain the output cluster sizes. So we propose a new grid-based clustering algorithm, as described in Figure 6.

The basic idea is as follows. First, let's denote $U = \{u_k, 1 \le k \le |U|\}$ as a set of users. For each user $u_k \in U$, we parse her GPS trajectories ($Traj^k$) and detect the stay points (S^k) from each trajectory by seeking some spatial regions where u_k spent a period over a certain threshold T_{thresh} and the distance between any two consecutive GPS points in it is less than D_{thresh}. For more details, please refer to our previous work [6]. After steps 1-3 in the algorithm, we have a stay point set $SP = \{S^k, 1 \le k \le |U|\}$ where each $S^k = \{s_1, s_2, \dots, s_N\}$ is the stay point set for user u_k.

```
Algorithm ExtractStayRegion($D_{thresh}, T_{thresh}, d$)
Input: A collection of GPS trajectories $\varphi = \{Traj^k, 1 \le k \le |U|\}$.
Output: A set of stay regions $R = \{r_i, 1 \le i \le m\}$, where $m = |R|$.
1.  Foreach $u_k \in U$ do
2.      $S^k$= StayPointDetection( $Traj^k$, $D_{thresh}$, $T_{thresh}$);
3.      $SP$.Add( $S^k$);                 // the collection of stay points
4.  $G$ = GridDivision( $d$ );            // divide the map into grids
5.  Foreach $g_i \in G$ do
6.      $g_i.sp = \{s_k | s_k \in SP$ within the region of $g_i\}$;
7.  For all $s_k \in SP$, set $s_k.regionID = -1$;     // initialization
8.  While (exists $s_k \in SP$ with $s_k.regionID = -1$) do
9.      Find $g_i$ with max $|g_i.sp|$ and unassigned to any stay region.
10.     $ng$ = GetNeighborGrids( $g_i$, $G$ );     // $ng$ is a set of grids
11.     $r = g_i \cup ng$;       // assign $g_i$ and $ng$ to a new stay region
12.     $(r.lat, r.lng)$ = GetCentroidCoordinates( $SP$, $r$ );
13.     $R$.Add( $r$ ) ;
14.     Foreach $s_j \in g_l.sp$ where $g_l \in r$ do
15.         $s_j.regionID = |R|$;       // assign region ID
16. Return $R$;
```

Figure 6. Grid-based clustering for stay region extraction

Second, we divide the map into grids (step 4), in order to constrain our output stay region to be limited in size. In particular, we set each grid as a square with width of $d/3$, where d is a parameter to constrain our output stay region size as no larger than $d \times d$ as shown later. After dividing the map into grids, we project all the detected stay points in these grids (steps 5-6), so that we have a set of grids $G = \{g_i, 1 \le i \le |G|\}$ with each $g_i \in G$ has its stay point set $g_i.sp$.

Third, we employ a greedy strategy to cluster grids (containing stay points). At each round (steps 8-15), we start with finding a grid g_i that is unassigned to any stay region yet and has the maximal number of stay points $|g_i.sp|$. Then, we will extract its 8 neighboring grids (i.e. consider a square shape with 3×3 grids and g_i in the center). The unassigned grids among these 8 neighboring grids, denoted as ng, are clustered with g_i to form a new stay region $r = g_i \cup ng$. Hence, all the stay points in g_i and ng are clustered into the stay region r. Note that at most there will be 3×3 grids clustered to a stay region, so we can constrain the extracted stay region size as $d \times d$. Finally, we calculate the centroids of all the stay points' latitude and longitude coordinates in r as r's coordinates. At last, we output a set of stay regions R.

3.2 Location-Activity Information Extraction

Based on stay region extraction, we can get a set of stay regions from the stay points. For each stay region r_i in the stay region set $R = \{r_i, 1 \le i \le m\}$, we can first extract the comments from the GPS data that attached to this stay region. After that, we parse the comments, which in general are texts, to get the activities. For example, if a comment mentions "delicious" or "restaurant", then it implies that the user had some foods or drinks at this location. Hence, we can have an activity of "food and drink" on this location. By parsing all the comments, we can get the counts of various activities on each stay region (location). Specifically, for a location i, we can have an n-dimensional count vector $\boldsymbol{c}_i = [c_{i1}, c_{i2}, \dots, c_{in}]$ for n activities, where each c_{ij} is the number of times when activity j performed at location i according to the comments. Denote the location-activity matrix as $X_{m\times n}$; then we can define its entries as:

$$X_{ij} = c_{ij}, \forall i = 1, \dots, m;\ j = 1, \dots, n \qquad (2)$$

Note that some locations may not have any comments, so their count vectors are zero vectors and the corresponding entries in the matrix X are zeros. However, when $X_{ij} = 0$, it doesn't mean that there is no possibility to perform activity j at location i. It is just because there is no comment that records that activity. So we treat all these entries equal to 0 as missing values for predictions.

3.3 Location-Feature Extraction

As discussed in the Section 2, we can use the POI category database to get the statistics (counts) of different POIs in an interested region. In particular, given a stay region $r_i \in R, 1 \le i \le m$, we will count the number of different POIs in an enclosing rectangle of the stay points in r_i, with the coordinates as [$r_i.lat$ - $d/2$, $r_i.lat + d/2$] × [$r_i.lng - d/2$, $r_i.lng + d/2$]. Here, d is the size parameter as introduced in Section 3.1. Therefore, the size of the enclosing rectangle is $d \times d$. Denote the count vector for a location i as $\boldsymbol{q}_i = [q_{i1}, q_{i2}, \dots, q_{il}]$ for l 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) [8] to obtain a location-feature matrix $Y_{m\times l}$. Specifically, we have each entry of Y as

$$Y_{ij} = \frac{q_{ij}}{\sum_{j=1}^{l} q_{ij}} \cdot \log \frac{|\{\boldsymbol{q}_i\}|}{|\{\boldsymbol{q}_i : q_{ij} > 0\}|}, \forall i = 1, \dots, m;\ j = 1, \dots, l, \qquad (3)$$

where $|\{\boldsymbol{q}_i\}|$ is the number of all the count vectors (i.e. number of locations), and $|\{\boldsymbol{q}_i: q_{ij} > 0\}|$ is the number of count vectors (i.e. locations) having non-zero j-th type POIs. In this way, we reasonably 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).

3.4 Activity-Activity Correlation Extraction

Knowing the correlations between the 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 correlations is to calculate them 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 senses and possibly reflected on the World Wide Web. To facilitate such common sense knowledge mining, we turn to Web search for help. In particular, for each pair of activities a_i and a_j, we put their names together as a query and submit it to some commercial search engine to get the Webpage hit counts. For example, for activities "food and drink" and "shopping", we generate a query "food and drink, shopping" and send it to Bing. Bing will then return 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" (30.3 million hits from Bing) is higher than that for "food and drink, sports and exercises" (7.56 million hits), showing that the correlations of "food and drink" with "shopping" is higher than that with "sports and exercise", coinciding with the common sense.

Based on such a method, we can then have an activity-activity matrix $Z_{n\times n}$, with each entry defined as

$$Z_{ij} = h_{ij}/h^*, \forall i = 1, \dots, n;\ j = 1, \dots, n, \qquad (4)$$

where h_{ij} is the hit count for activity i and activity j based on some search engine. $h^* = argmax\ h_{ij},\ \forall i, j$ is the maximal hit count among all the hit counts for each pair of activities.

4. COLLABORATIVE LOCATION AND ACTIVITY RECOMMENDATIONS

After the data modeling, we have the location-activity, location-feature and activity-activity matrices. As the location-activity matrix is incomplete with many missing entries, our objective is to fill those missing entries so as to get a full location-activity matrix for location and activity recommendations. Since the location-activity matrix is very sparse, we try to borrow some more information from location-feature and activity-activity matrices for prediction based on collaborative filtering.

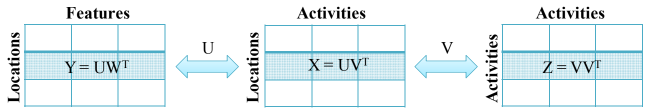


Figure 7. Demonstration of our model

Figure 7 demonstrates the main idea of our model based on collective matrix factorization. Given the location-activity matrix $X_{m\times n}$, we decompose it by low-rank approximation as a product of two matrices $U_{m\times k}$ and $V_{n\times k}$ (the superscript "T" for $V_{n\times k}^T$ denotes the matrix transpose), where $k < n$. It shares the location information through sharing matrix $U_{m\times k}$ with the location-feature matrix $Y_{m\times l}$, which is decomposed as a product of matrices $U_{m\times k}$ and $W_{l\times k}$. Similarly, the location-activity matrix shares the activity information through sharing matrix $V_{n\times k}$ with the activity-activity matrix $Z_{n\times n}$, which is decomposed as a self product of $V_{n\times k}$. Hence, we put forward a collective matrix factorization model and formulate our objective function as:

$$L(U,V,W) = \frac{1}{2} \| I \circ (X - UV^T) \|_F^2 + \frac{\lambda_1}{2} \| Y - UW^T \|_F^2 + \frac{\lambda_2}{2} \| Z - VV^T \|_F^2 + \frac{\lambda_3}{2} (\| U \|_F^2 + \| V \|_F^2 + \| W \|_F^2), \quad (5)$$

where $\|\cdot\|_F$ denotes the Frobenius norm. I is an indicator matrix with its entry $I_{ij} = 0$ if X_{ij} is missing, $I_{ij} = 1$ otherwise. The operator "$\circ$" denotes the entry-wise product. As shown in the Figure 7 and the objective function, we aim to propagate the information among $X_{m\times n}$, $Y_{m\times l}$ and $Z_{n\times n}$, by requiring them to share some low-rank matrices $U_{m\times k}$ and $V_{n\times k}$. The first three terms in the objective function (5) control the loss in matrix factorization, and 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 $U_{m\times k}$, $V_{n\times k}$ and $W_{l\times k}$, and we cannot get closed-form solutions for minimizing the objective function. Therefore, we will turn to some numerical method such as gradient descent to get the local optimal solutions. Specifically, we have the gradients (denoted as ∇) for each variable as

$$\begin{aligned} \nabla_U L &= [I \circ (UV^T - X)]V + \lambda_1 (UW^T - Y)W + \lambda_3 U, \\ \nabla_V L &= [I \circ (UV^T - X)]^T U + 2\lambda_2 (VV^T - Z)V + \lambda_3 V, \quad (6) \\ \nabla_W L &= \lambda_1 (UW^T - Y)^T U + \lambda_3 W. \end{aligned}$$

After having the gradients, we can use gradient descent to iteratively minimize the objective function. The details of the algorithm are given in Figure 8.

```
Algorithm CLAR
Input: Incomplete location-activity matrix $X_{m\times n}$, location-feature
matrix $Y_{m\times l}$ and activity-activity matrix $Z_{n\times n}$.
Output: Complete location-activity matrix $X_{m\times n}$.
1. $t = 1$;
2. While ($t < T$ and $L_t - L_{t+1} > \epsilon$) do        // T is #(max iterations)
3.     Get the gradients $\nabla_{U_t}$, $\nabla_{V_t}$ and $\nabla_{W_t}$by Eq.(6);
4.     $\gamma = 1$;
5.     While ($L(U_t - \gamma\nabla_{U_t}, V_t - \gamma\nabla_{V_t}, V_t - \gamma\nabla_{V_t}) \geq L(U_t, V_t, W_t)$) do
6.         $\gamma = \gamma/2$;     // search for the maximal step size
7.     $U_{t+1} = U_t - \gamma\nabla_{U_t}$, $V_{t+1} = V_t - \gamma\nabla_{V_t}$ and $W_{t+1} = W_t - \gamma\nabla_{W_t}$;
8.     $t = t + 1$;
9. Return X;
```

Figure 8. Algorithm description for our model

After having the complete location-activity matrix $X_{m\times n}$, for a user query of some location, we can look up the rows of $X_{m\times n}$. If this location exists in our system (i.e. the location coordinates fall in some stay region), for example, the *i*-th row of $X_{m\times n}$, we rank the *i*-th row's values in a descending order and return a list of corresponding activities for activity commendation. For example, we search "Bird's Nest" in our system and find it matched with 10th row (i.e. 10th location) of $X_{m\times n}$, then we will extract the 10th row's ratings, e.g. *x*=[2, 3, 4, 5, 1], where each entry denotes the rating for an activity. Assume from left to right in x, the activities are "Food", "Shopping", "Sports", "Tourism" and "Movie", then we will recommend a ranking list of activities with "Tourism" > "Sports" > "Shopping" > "Food" > "Movie". Similarly, for location recommendation, given a user query of some activity, we look up the columns of $X_{m\times n}$. If this activity is matched in our system, for example, the *j*-th column of $X_{m\times n}$, we rank the *j*-th column's values in a descending order and return a list of the top *N* corresponding locations for location commendation.

5. EXPERIMENTS

In this section, we will first present the experimental settings. Second, we will introduce the evaluation approaches. Third, we will deliver some major results followed by some discussions.

5.1 Settings

5.1.1 GPS Users, Devices and Data

In total, we have 162 users (61 females and 101 males, and more statistics is shown in Figure 9) carrying the GPS devices to record their outdoor trajectories from April 2007 to Oct. 2009.

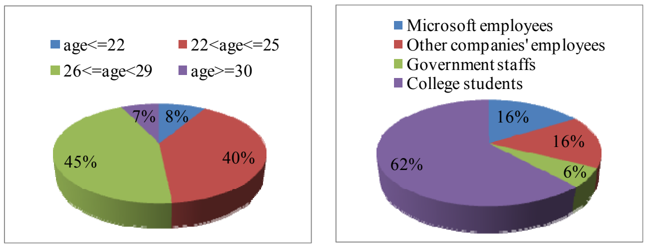
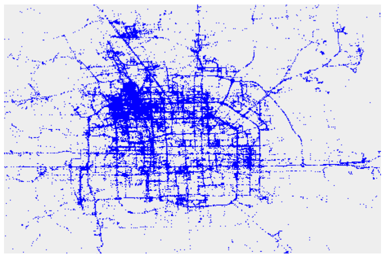


Figure 9. GPS user statistics

Figure 10(A) shows the GPS devices used to collect data. They are comprised of stand-alone GPS receivers and GPS phones. In general, the sampling rate for GPS devices is set as two seconds. The GPS logs were collected in China, as well as a few cities in the USA, 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 12,765 GPS trajectories with total GPS points number over 3,980,320 and total trajectory length over 139,310 kilometers. We have totally 530 comments. To make sure that we recommend useful locations and activities, we also remove some GPS points for work and homes. The data distribution in Beijing is shown in Figure 10(B). To protect the users' privacy, we use these data anonymously.



(A) GPS devices (B) Data distribution in Beijing

Figure 10. GPS devices and data distribution

We defined 5 activities to recommend from the GPS data. As shown in Table 1, these 5 activities basically cover people's daily routines. Therefore, *n*=5 for the matrices *X* and *Z* in Eq.(5).

Table 1. 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.

5.1.2 Parameter Selection

Data processing parameters. In this experiment, to obtain stay points from raw GPS data, we follow our previous work [9] to set T_{threh} as 20 minutes and D_{threh} as 200 meters for stay point detection. To extract stay regions from stay points, we tentatively set the stay region size as 300×300 square meters, i.e. d = 300, and we will study its impact in Section 5.3.2.

Model parameters. Our model has 3 parameters: λ_1, λ_2 and λ_3, where λ_1 and λ_2 control the contributions of location features and activity correlations respectively, and we will study their impacts in Section 5.3.1. λ_3 controls the regularization term, and we set it as 10 through all our experiments. As our model is based on low-rank matrix factorization, we set the rank *k*=3 for the matrices *U*, *V* and *W* in Eq.(5).

5.2 Evaluation Methodology

To evaluate our recommendation system, we invited 5 subjects who are familiar with Beijing, to individually use our system and provide the feedbacks. For activity recommendation, we asked the subjects to evaluate the top 5 recommended activities on the top 20 popular locations according to the GPS logs; and for location recommendation, we asked them to evaluate top 10 recommended locations. Our system's user interface is shown in Figure 4, where users can provide ratings to the recommended locations/activities.

Table 2. Rating criteria for locations and activities

Ratings	Explanations
3	I'd like to visit this location / do this activity
2	I'd like to visit this location if passing by / do this activity if time spared
1	I have no feeling to visit this location / do this activity
0	This location does not deserve to visit / this activity is not suitable to do there.

In Table 2, we list the rating criteria. To get the ground truths for evaluation, we aggregate all the subjects' feedbacks to get an ideal ranking list. As our recommendations are based on ranking results, we employ normalized discounted cumulative gain (nDCG) [8] to measure our retrieved location/activity list. nDCG is commonly used in information retrieval to measure the search engine's performance. A higher nDCG value to a list of search results means that, the highly relevant items appearing earlier (with higher ranks) in the result list. In particular, nDCG[p], or referred as nDCG@p, measures the relevance of top p results:

$$nDCG[p] = \frac{DCG[p]}{IDCG[p]},\ DCG[p] = rel_1 + \sum_{i=2}^{p} \frac{rel_i}{\log_2 i},$$

where $IDCG[p]$ is the $DCG[p]$ value of ideal ranking list. rel_i is a relevance value. nDCG ranges from 0 to 1. The higher nDCG is, the better a ranking result list is. For example, given a ranking list of 4 items with relevance as <1,3,0,2>, the nDCG@4 is

$$nDCG[4] = \frac{1+3/log_2 2+0+2/log_2 4}{3+2/log_2 2+1/log_2 3} = 0.89.$$

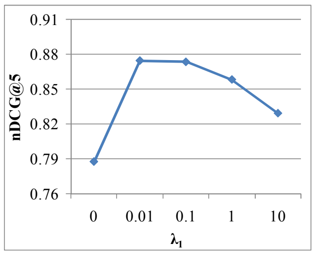
5.3 Results and Discussions

In this section, we will first study our system's performances under different model parameters. Then, we will employ two baselines for comparison under several settings. Finally, we will give some more insights to our model.

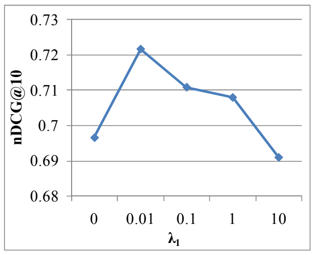
5.3.1 System performance

5.3.1.1 Impact of the location feature information

The parameter λ_1 controls the contribution of the location feature information to the objective function (5). To study the impact of this information, we vary the value of λ_1 and plot our model's performances with the average nDCG values over all the subjects in Figure 11. In this study, we fix λ_2=200, which is in the magnitude of division of the location-feature matrix size by the activity-activity matrix size. In this way, we make sure that both location features and activity correlations can contribute to the objective function. We use nDCG@5 to evaluate the results for activity recommendation and nDCG@10 for location recommendation in this paper if without specific references.



(A) activity recommendation



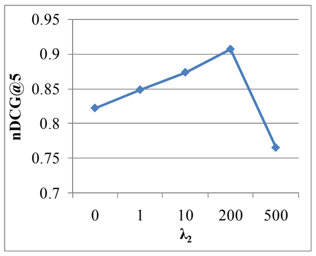
(B) location recommendation.

Figure 11. Impact of the location features

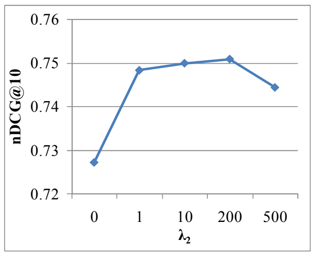
As shown in Figure 11(A), the model's performance first increases and later decreases as λ_1 increases. This is because when λ_1 is too small, the model cannot fully utilize the information from the location features to understand the location functionalities and the connections among the locations. When λ_1 is too large, the location feature information will dominate the objective function (5), thus overwhelming the activity information from the location-activity matrix X and activity-activity matrix Z. It will cause some problem; for example, given two locations i and j with similar location features, if location i has no rating for "food and drink" but has ratings for "movie and shows", according to location similarity, we cannot recommend to users to try "food and drink" at location j although usually there are some nice restaurants near the movie theatres. In Figure 11(B), we observe the similar pattern for location recommendation. When the location-activity ratings are not well inferred with too small or too largeλ_1, the recommended location list also may miss some interesting places. Note that, when λ_1=0, the model equals to only exploiting one additional information source, i.e. the activity-activity correlation, to help recommendation. As the performance at λ_1=0 can be lower than the performance at λ_1>0 (e.g. λ_1=0.1), we demonstrate the benefit of using both additional information.

5.3.1.2 Impact of the activity correlation information

We also study the impact of parameter λ_2, which controls the contribution of the activity correlation information to the objective function (5). We vary the value of λ_2 and plot our model's performances in Figure 12. In this study, we fix λ_1=0.1 according to the previous study in Figure 11.



(A) activity recommendation



(B) location recommendation

Figure 12. Impact of the activity correlations

As shown in Figure 12(A), similarly we observe the model's performance first increasing and later decreasing as λ_2 increases. When λ_2 is too small, the activity correlation information cannot contribute much to the objective function. When λ_2 is too large, the activity correlations will dominate the objective function, so that for a location it will recommend the activities mostly based the correlation values while not fully considering whether such a location is suitable for some activity. For example, if a location has some ratings for "food and drink", then with too large λ_2, the model will also recommend the user to see "movie and shows" without carefully considering whether this location has theatre or not. In Figure 12(B), we can observe the similar pattern for location recommendation. Note that, when λ_2=0, the model equals to only exploiting one additional information source for location features. Again, as the performance at λ_2=0 can be lower than the performance at λ_2>0 (e.g. λ_2=200), we demonstrate the benefit of exploiting both additional information sources.

5.3.2 Investigation into Our System

5.3.2.1 Comparison with baselines

We employ two baselines: single collaborative filtering (*SCF*) and unifying collaborative filtering (*UCF*). In *SCF*, we only use the incomplete location-activity matrix as input for collaborative filtering. We employ the popular low-rank matrix factorization approach to accomplish such a collaborative filtering task [18]. In particular, *SCF* aims to solve an singular value decomposition problem by $\min J(U,V) = \| I \circ (X - UV^T) \|_F^2$, where X denotes the incomplete location-activity matrix, U and V are the low-rank matrices, I is the indicator matrix same with Eq.(5). It can be seen that this optimization problem equals to the case when our objective function (5) has both λ_1 and λ_2 as zeros. We employ this baseline to show that with limited number of comments (and thus sparse in location-activity matrix), the recommendation results may not be satisfying. So we can validate our motivation to use additional information sources to help improve recommendation.

We also follow [10] to provide a solution, *UCF,* which can use the additional information sources for unifying collaborative filtering. In *UCF*, for each missing entry in the location-activity matrix, it will extract a set of top *N* similar locations and top *N* similar activities, and then use the ratings for these users over these items in a probabilistic way to calculate a value for the missing entry. After all the missing entries are filled in the location-activity matrix, similar ranking strategy with our system can be used to output the location and activity ranking list for recommendations. We use this baseline to testify the effectiveness of our model over other collaborative filtering methods given the same inputs.

Table 3. Comparisons under different p-values for nDCG@p

	Activity Recommend.		Location Recommend.		
	p=3	p=5	p=5	p=8	p=10
CLAR	**0.83±0.04**	**0.91±0.03**	**0.84±0.06**	**0.84±0.04**	**0.86±0.04**
UCF	0.72±0.06	0.87±0.03	0.76±0.03	0.74±0.03	0.75±0.03
SCF	0.70±0.07	0.84±0.05	0.63±0.08	0.62±0.07	0.63±0.06

In Table 3, we report the performances of our model and other two baselines for both activity and location recommendations. We vary the *p*-values for nDCG@*p* to extensively evaluate the systems' performances. The entry value in Table 3 denotes the mean and standard deviation of the nDCG values. As shown in the table, our model CLAR consistently outperforms the two baselines under different measurements. We also conduct the t-test over the results and find our results are significantly better than the baselines' results (one-tailed test $p_1<0.01$, two-tailed test $p_2<0.01$) in both location and activity recommendations. Both our CLAR and *UCF* can outperform *SCF* due to using more information. Besides, our CLAR can outperform *UCF* because in *UCF*, the information flow is in a single direction from location features and activity correlations while our CLAR enables the information flow in both directions. In other words, in *UCF*, the location similarities and activity similarities are learned from the location features and activity correlations; then they are passed to the location-activity matrix for collaborative filtering. This collaborative filtering does not have further feedback to the location-features and activity correlations. So, if the similarities learned from this additional information are not accurate, there is no second chance to refine. In contrast, in our CLAR, we put the location-activity matrix and the two pieces of additional information together in an objective function for optimization, so that we can have the feedback from the matrix factorization in location-activity matrix to the location-feature matrix and activity-activity correlation matrix. In this way, our CLAR can have bi-directional information flows and thus outperform *UCF*.

5.3.2.2 Impact of the stay region size

We also study the impact of stay region size in recommendation. As discussed above, in recommendations, we may prefer smaller stay region size so that the users can easily find what she wants in the recommended location. Therefore, we vary the stay region size by varying the region width *d* from 200 to at most 500 (*d*=500 means that the stay region size is 500×500 square meters).

As shown in Table 4, as the stay region size increases, the number of stay regions extracted by grid-based clustering (shown in Figure 6) decreases. Our CLAR model consistently outperforms the two baselines *UCF* and *SCF*. We also conduct the t-test and find our results are better than the baselines' results (one-tailed test $p_1<0.05$, two-tailed test $p_2<0.05$) in both location and activity recommendations. When *d*=300, our CLAR works the best, showing that too small region size may make the extracted stay regions' location features insufficient to represent the location functionalities and too large region size may lead to difficulty in finding interested points of interests from a big area.

Table 4. Impact of stay region size

	#(stay region)	Activity Recommend.			Location Recommend.		
		CLAR	UCF	SCF	CLAR	UCF	SCF
d=200	3329	**0.86 ±0.02**	0.85 ±0.02	0.83 ±0.02	**0.82 ±0.03**	0.72 ±0.03	0.58 ±0.05
d=300	2503	**0.91 ±0.03**	0.87 ±0.03	0.84 ±0.05	**0.86 ±0.04**	0.75 ±0.03	0.63 ±0.06
d=500	1696	**0.86 ±0.01**	0.81 ±0.03	0.83 ±0.02	**0.86 ±0.03**	0.74 ±0.03	0.67 ±0.02

5.3.2.3 Impact of the user number

As the GPS devices become popular, we will have more and more users and accumulate such GPS data on the Web as time goes by. We study the impact of user numbers so as to see whether our system can handle the data well.

Table 5. Impact of user number

	#(stay point)	Running Time (ms)	Activity Recommend.	Location Recommend.
#user=50	3895	5780.15	0.84±0.04	0.75±0.03
#user=100	8039	10828.45	0.88±0.03	0.89±0.02
#user=162	12656	15053.6	**0.90±0.03**	**0.91±0.03**

As the user number increases, the GPS data size increases and thus the number of stay points also increases (though it might not be the case when the user number is sufficiently large). As shown in Table 5, the running time for our CLAR model is almost linear to the number of stay points. This is because the computational complexity of our CLAR model is linear to the number of stay points. Consider the algorithm for our CLAR model in Figure 8. Given the input matrices $X_{m\times n}$, $Y_{m\times l}$, $Z_{n\times n}$ and their low-rank factorized matrices $U_{m\times k}$, $V_{n\times k}$, $W_{l\times k}$, we have the computational complexity of evaluating the objective function (5) is: $m\times k\times n+m\times k\times l+n\times k\times n+(m\times n+n\times k+l\times k)$, which is $O(m)$ since *n*, *l* and *k* are much smaller than *m* (e.g. in our case with 162 users, *m*=12656, *n*=5, *l*=13, *k*=3). Similarly, we can have the computational complexity for the gradients as $O(m)$. As our algorithm has an iteration limit and in practice it converges fast (in less than 300 iterations), the whole computational complexity for our model is linear to the number of stay points $O(m)$. Hence, our model can be quite efficient. From Table 5, we also observe that as the user number increases, there are more GPS data and thus we can keep improving the system's performance.

5.3.3 Discussions

5.3.3.1 Impact of the location types to activity recommendation

Is our system doing equally well on activity recommendation for different types of locations? We summarize the experimental results for the setting with d=300 and #user=162 to answer this question in Figure 13. As can be seen from the figure, for the 20 most popular locations, our system works the best on the locations that are in the type of "food and sports area", and the worst on the locations that are in the type of "shopping and movie area" (here we aggregate the user evaluations and pick the top 2 activities as the location types). This is because the activity "food and drink" happens more often in our daily life; and it's also more likely to have many restaurant POIs in the location feature for predicting this activity. For "sports" areas, the location features can capture the location functionality by detecting the parks and stadiums. For "tourism" areas, there are more comments from the GPS users, so that the prediction on such areas can be comparatively accurate. For the "shopping & movie" area, the activity recommendation results are not as good as the other areas, because there are fewer comments from the GPS users on these activities and thus fewer ratings in collaborative filtering. Besides, such areas are usually also suitable for food hunting and sometimes tourism, so that they are overwhelmed by the recommendations to food and tourism.

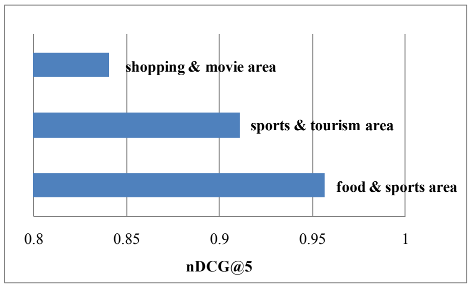


Figure 13. Impact of location types to activity recommend.

5.3.3.2 Impact of the activity types to location recommendation

We also ask the question whether our system does equally well on location recommendation for different activity types? Based on the same setting with previous section, we summarize the location recommendation results on each activity type in Figure 14.

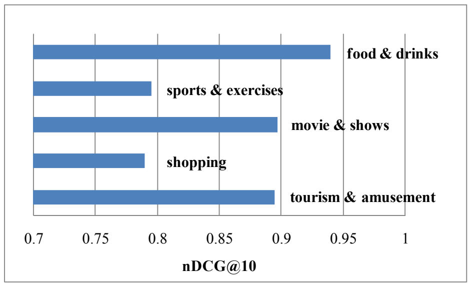


Figure 14. Impact of activity types to location recommend.

As expected, for activity "food and drinks" which more often happens, and activity "tourism and amusement" which has more user comments, the recommendation results are quite satisfying. For "movie and shows", the results are still good. For "shopping", the performance is worse due to less user comments in modeling the location-activity matrix for collaborative filtering. For "sports and exercises", the performance is also worse than other activities; and an interesting observation is that, in Figure 13, our system usually performs well on the "sports" areas. Is there something wrong? By analyzing the data, we find that this is reasonable; because in our system the locations with more comments are more likely to be recommended (i.e. the higher ratings on other activities can propagate to the activity "sports"), but most of these locations are related to food hunting and tourism which are loosely connected with "sports". As a result, the location recommendation for the activity "sports" is worse than the others.

5.3.3.3 Prediction for new locations and activities

Our system is based on some GPS data which is limited in size. Therefore, there could be some locations that we do not see in the existing GPS dataset. Similarly, we also only define 5 main activities, what if the user wants to get recommendations for some more-detailed activities, such as "Thai food" instead of general "food"? One possible solution could be relying on the data accumulation on the Web. As the GPS devices become popular, there can be more and more GPS data related to more detailed activities in people's daily life. Once we have these data, we can keep updating our system. Since our model's computational complexity is linear to the number of GPS stay points (i.e. the data size), such updates could be easy. Another possible solution is to get such location-activity information from the Web. As there are blogs describing such information (e.g. travel logs), we may mine such knowledge from the Web to enhance our system. However, considering that the blog contents can be quite noisy, it's not clear how much it helps. We may leave it as our future study.

6. RELATED WORK

6.1 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 [12], a mobile application framework, which enables a mobile phone user to query the geo-coded Wikipedia articles for landmarks in vicinity, is presented. In [13], 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 [14], a CityVoyager system is developed to recommend shops. It collects the users' shop visiting 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 [15], 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 [16], a Geowhiz system, which uses a user-based collaborative filtering algorithm to recommend restaurants, is proposed. In [9], 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 in modeling only one type of location for recommendations, our system is capable to handle various types of locations. That is, we can recommend locations not only for foods and drinks but also for shopping, etc.

6.2 Activity Recommendation

Activity recommendation is a pretty new research issue with little research on it so far. 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 activity study focuses on how to recognize an activity from sensor data by ubiquitous computing [21]. For example, in [5], based on GPS data, a hierarchical conditional random field model is used to recognize whether a user is at work, or sleeping at home, or taking leisure, or visiting friend, etc. In [17], activities of daily living such as brushing teeth or making a snack in indoor environment are recognized by using RFID sensors. Some object use common sense knowledge is extracted from Web to help training a recognition model in an unsupervised way. In contrast, rather than recognize the activity for an individual user in real time, we aim to do mining over the users' activity histories (i.e. GPS logs) and recommend what a user can do on some location.

7. CONCLUSION

In this paper, we studied how to mine knowledge from the real-world GPS data to answer two common questions in our daily life. 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? This question corresponds to activity recommendation. We show that these two questions are inherently related, as they can be seen as a ranking problem over a location-activity rating matrix. Because the location-activity matrix is very sparse in practice, we proposed to exploit other information, including the location features and the activity-activity correlations from various information sources, to enhance the performance. We provided a collaborative filtering approach based on collective matrix factorization to take these information sources as inputs and train a location and activity recommender. Both PC and hand-held device users can access our recommender through the Web to get recommendations for better trip planning, etc. We evaluated our system on a large GPS dataset, and showed 7% improvement on activity recommendation and over 20% improvement on location recommendation over the simple baseline without exploiting any additional information.

In the future, we will consider more information, such as user features, to further enhance the performance. Our current system is for general recommendations; if we have the user features, we may be able to personalize our recommendation system so as to better satisfy the user's information needs. Besides, we may also use the user features to establish a social network among the users so that the experiences from friend (similar) users can contribute more in retrieving recommendation results.

8. ACKNOWLEDGEMENTS

This research was supported by grants from Microsoft Research Asia MRA08/09.EG03 and Hong Kong-CERG/China-NSFC N_HKUST624/09. The authors would like to thank Yukun Chen for his great help in data processing. The authors would also thank the reviewers for their helpful comments and suggestions.

9. REFERENCES

[1] Bikely: http://www.bikely.com/.

[2] GPS route exchange forum: http://www.gpsxchange.com/.

[3] Liao L., Fox D. and Kautz H. Learning and inferring transportation routines. In Artificial Intelligence 171(5-6): 311-331 (2007).

[4] Zheng Y., Liu L., Wang L. and Xie X. Learning transportation modes from raw GPS data for geographic applications on the Web. In Proc. of the 17th Intl. Conf. on World Wide Web (Beijing, China, 2008), ACM Press: 247-256.

[5] Liao L., Fox D. and Kautz H. Location-based activity recognition. In Proc. of Advances in Neural information Processing Systems (Vancouver, Canada, 2005).

[6] Li Q., Zheng Y., Xie X., Chen Y., Liu W. and Ma W. Mining user similarity based on location history. In Proc. of the 16th Intl. Conf. on Advances in geographic info. system (Santa Ana, USA, 2008), ACM Press: 1-10.

[7] Microsoft GeoLife Project: http://research.microsoft.com/en-us/projects/geolife/

[8] Manning D., Raghavan P. and Schütze H. Introduction to Information Retrieval. Cambridge University Press, 2008.

[9] Zheng Y., Zhang L., Xie X. and Ma W. Mining interesting locations and travel sequences from GPS trajectories. In Proc. of the 18th Intl. Conf. on World Wide Web (Madrid Spain, 2009), ACM Press: 791-800.

[10] Wang J., de Vries A.P. and Reinders M.J.T. Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In Proc. of the 29th ACM SIGIR conference on Research and development in information retrieval (Seattle USA, 2006), ACM Press: 501-508.

[11] Singh A.P. and Gordon G.J. Relational learning via collective matrix factorization. In Proc. of the 14th SIGKDD Intl. Conf. on Knowledge discovery and data mining (Las Vegas USA, 2008), ACM Press: 650-658.

[12] Simon R. and Frőhlich P. A mobile application framework for the geospatial Web. In Proc. of the 16th Intl. Conf. on World Wide Web (Canada, 2007). ACM Press: 381-390.

[13] Abowd G.D., Atkeson C.G., Hong J., Long S., Kooper R. and Pinkerton M. Cyberguide: a mobile context-aware tour guide. In Wireless Network, 3(1997), 421-433.

[14] Takeuchi Y. and Sugimoto M. CityVoyager: an outdoor recommendation system based on user location history. In Proc. of Ubiquitous Intelligence and Computing (Berlin Germany, 2006), Springer Press: 625-636.

[15] Park H., Hong H. and Cho B. Location-based recommendation system using Bayesian user's preference model in mobile devices. In Proc. of Ubiquitous Intelligence and Computing (Hong Kong, 2007). Springer Press: 1130-1139.

[16] Horozov T., Narasimhan N. and Vasudevan V. Using Location for Personalized POI Recommendations in Mobile Environments. In Proc. of the Intl. Symposium on App. on Internet (Phoenix, USA, 2006), IEEE Press: 124-129.

[17] Wyatt D., Philipose M. and Choudhury T. Unsupervised activity recognition using automatically mined common sense. In Proc. of the 20th national conference on Artificial intelligence (Pittsburgh, USA, 2005), AAAI Press: 21-27.

[18] Srebro N. and Jaakkola T. Weighted low-rank approximations. In Proc. of the 21st Intl. Conf. on Machine Learning (Washington DC, USA, 2003), AAAI Press 720-727.

[19] Ankerst M., Breunig M., Kriegel P. and Sander J. OPTICS: ordering points to identify the clustering structure. In Proc. of SIGMOD (New York, USA, 1999), ACM Press: 49-60.

[20] Chen Y., Fabbrizio D., Gibbon D., Jana R., Jora S., Renger B. and Wei B. GeoTV: navigating geocoded rss to create an iptv experience. In Proc. of the 16th intl. conf. on World Wide Web (Banff Canada, 2007). ACM Press: 1323-1324.

[21] Zheng V., Hu D. and Yang Q. Cross-domain activity recognition. In Proc. of the 11th Intl. Conf. on Ubiquitous Computing (Orlando, USA, 2009). ACM Press: 61-70.

[22] Zheng Y., Zhang L. and Xie X. Recommending friends and locations based on individual location history. In ACM Transaction on the Web (2010).