Noname manuscript No.
(will be inserted by the editor)

Hidden Location Prediction using Check-in Patterns in Location Based Social Networks

Pramit Mazumdar · Bidyut Kr. Patra · Korra Sathya Babu · Russell Lock

Received: date / Accepted: date

Abstract Check-in facility in a Location Based Social Network (LBSN) enables people to share location information as well as real life activities. Analysing these historical series of check-ins to predict the future locations to be visited has been very popular in the research community. However, it has been found that people do not intend to share the privately visited locations and activities in a LBSN. Research into extrapolating unchecked locations from historical data is limited. Knowledge of hidden locations can have a wide range of benefits to society. It may help the investigating agencies in identifying possible places visited by a suspect, a marketing company in selecting potential customers for targeted marketing, for medical representatives in identifying areas for disease prevention and containment, etc. In this paper, we propose an Associative Location Prediction Model (ALPM), which infers privately visited unchecked locations from a published user trajectory. The proposed ALPM explores the association between a user's checked-in data, the Hidden Markov Model and proximal locations around a published check-in for predicting the unchecked or hidden locations. We evaluate ALPM on real-world Gowalla LBSN dataset for the users residing in Beijing, China. Experimental results show that the proposed model outperforms the existing state of the art work in literature.

Keywords Location Prediction · Location Based Social Networks · Ranking · Similarity Measure · Trajectory Analysis

1 Introduction

The rapid advancement of technology has made possible handheld devices to accomplish many tasks which were previously restricted to desktop applications.

P. Mazumdar (✉) · B. Kr. Patra · K. S. Babu
National Institute of Technology Rourkela, Odisha, India
E-mail: {512cs1011,patrabk,ksathyababu}@nitrkl.ac.in

R. Lock
Loughborough University, Leicestershire, United Kingdom
E-mail: r.lock@lboro.ac.uk

These days mobile devices like smart phones and tablet PCs provide real time GPS data which collects vast spatio-temporal information of user movements. Many applications including Bikely[1], Gpsshare[2], Gpsxchange[3], etc. have been developed to record user's travel experiences through GPS logs, and subsequently publish their GPS information on the internet. In 2009, this notion was further extended into a new breed of social networking named as the Location Based Social Network (LBSN). Some popular examples of LBSNs are Foursquare, Gowalla, Brightkite, etc.

LBSN offers a facility called check-in by which a user can share spatio-temporal information (position and time), perform various activities and maintain social relationships with other users. Check-ins are performed in real time and hence the trajectories obtained from them represent the physical movement of a user. Thus, check-in services in LBSNs bridge gap between the online social world and the physical world. However, it has been found that people are reluctant to share privately visited locations in their published trajectories. Revealing the unchecked locations can help advertisers to identify users for target marketing. It can also help investigating bodies to find traces of a suspect from the published trajectory. The problem of location prediction has become popular in recent years [1–3].

Analysis of large spatio-temporal datasets show that check-ins tend to be more concentrated around a few key places of interest. The semantic meaning of these locations and their geographical positions are used as inputs to the location prediction problem. Moreover, similar check-ins performed by socially linked individuals may also help in location prediction. Researchers have thus explored the combination of social relationships and spatio-temporal data to predict locations [4–6]. It is to be noted that the existing works as described earlier, mainly focus on the problem of next location or destination prediction by analysing the trajectory of users. However, this paper focuses on hidden location prediction. Privately visited locations which are not available in a user's published trajectory are the hidden or unchecked locations. Therefore, considering a published trajectory $\langle l_a \rightarrow l_b \rightarrow l_c \rightarrow l_d \rangle$, the problem of predicting unchecked locations for any two consecutive pair of check-ins say, $\langle l_b \rightarrow l_c \rangle$ is termed as the hidden location prediction problem. Although it utilizes historical check-in data, the target location in our case is different from the next location or destination location prediction problem.

In this paper, the proposed model explores a data mining technique to infer hidden locations from the published trajectory of users. Association rule mining is used in the proposed approach to extract how frequently consecutive check-in pairs occur. Mining the sequence of user check-ins helps to find associations between a pair of consecutive check-ins. We present a unified view of the interplay between associated check-ins, users with similar trajectories and the geographically proximal locations. Experimental results show that this unsupervised learning technique is very effective in identifying unchecked locations from users' published trajectory.

[1] www.bikely.com
[2] www.gpsshare.com
[3] www.gpsxchange.com

Further, for hidden location prediction we also consider the published trajectories as a markov model, with sequence of observed and unobserved states or locations. For any pair of candidate check-ins, our model selects N number of hidden locations. Hence, a novel ranking technique is introduced for arranging the predicted locations. It achieves better results when compared with other existing ranking techniques like the Bayesian inference [15,23] and variants of collaborative filtering [25,26]. The contributions of this paper are summarized as follow.

1. An algorithm is devised for identifying the consecutive location pairs which are potential candidates for hidden location prediction.
2. Similar users are computed using a proposed similarity metric, which considers the sequence of check-ins performed.
3. A new location prediction model for LBSNs is developed to predict the unchecked or hidden locations from a user trajectory.
4. A novel ranking metric called Entropy_Distance is proposed for ranking the predicted hidden locations.
5. The model is evaluated on a real world dataset using standard statistical measures. Results show that our model outperforms the existing state of the art technique.

The rest of the paper is organized as follows. In Section 2, we survey related works on spatio-temporal data analysis, location prediction and various techniques to identify similar users. Motivation for this work, along with the problem of identifying hidden locations is given in Section 3. The architecture of the proposed Associative Location Prediction Model (ALPM) with the subsequent algorithms and explanations are provided in Section 4. In Section 5, the performance of ALPM is evaluated using Gowalla check-in dataset on the active users of Beijing. Experimental results and related discussions are provided in this section. Section 6 includes the limitations of the current work and discussions on possible future directions of this work. Finally, we conclude our work in Section 7.

2 Related Work

The convenience of spatio-temporal data collection technology has led to different types of analysis on user behaviours in social networks. These can be categorized into spatio-temporal data analysis, identifying hidden social ties, finding similar users, location prediction, etc.

Spatio-temporal data can be obtained from moving object (e.g. taxi, bird) trajectories recorded by GPS devices, social events (e.g. microblogs, posts) with location tag and timestamps, environment monitoring (e.g. weather forecasting), etc. [7]. Hidden social ties in online social networks refers to identifying relationships between users, not having direct connection in the social graph [8]. The works in [1,9] study user mobility patterns for predicting social links between users. Contextual information such as users' geographic movement, pattern of friendship formation and temporal dynamics are major factors that govern mobility of a user in a social network.

Finding similarity between users based on location histories help one to retrieve information with high relevance. Research on finding similar users in a social network requires a clear understanding of users' behaviour over time across various social-activities. Approaches proposed in [5, 10, 11] analyse user activities in social networks to find similar users. Li *et al.* propose a hierarchical graph-based similarity measurement model (HGSM), which clusters the visited locations of a user into a hierarchical graph [10]. They measure similarity as the maximum common length of a path traversed sequentially. However, the geographic distance between locations and the semantic meaning of popular locations are not considered. Scellato *et al.* [5] propose two models, *Geo* and *Social* for computing similarity between users by observing the geographic distance between checked-in locations. Lee *et al.* [11] use the semantics of the location to observe intention and interest of a user. They construct a location category hierarchy graph to represent the semantics of locations. It consists of nodes representing the locations visited by a user and category nodes representing the category of the visited locations. To identify the similar users, first they compute the significant score of each visited location with respect to a user. The significant score is the ratio between the number of times a user visited a location to the total number of visits observed for that user. Subsequently, the top-N locations with high significant score are identified and a location category hierarchy tree is generated. Finally, the similarity score between a pair of users is computed from the multiple propagation rate and the significant score at each location node in the location category hierarchy graph.

Backstrom *et al.* [4] identify the future location of a user from published check-ins and the locations visited by their social ties. They show that the distance between the current location of two users play an important role in making friendship. People living within shorter distances have a higher probability of social ties. These social ties are ranked on the basis of their distance from the concerned users. The locations visited by the close friends are finally utilised for location prediction. However, temporal information is not considered in their work. Cho *et al.* [12] consider temporal information of each check-in for predicting the next location to be visited. The day-to-day movement patterns of an active user along with their social ties are combined for analysis. Their proposed models (PMM and PSMM) are based upon mobility of users in the social network. Their analysis reveals that about 50% of user movements are governed by their social relationships. However, the research does not emphasize on identification of the hidden locations between two consecutive check-ins. Sadilek *et al.* [13] consider the content of messages, patterns in social link formation and the checked-in locations for more precise predictions. Their proposed system (FLAP) addresses the problem of link and location prediction in a social network. However, they do not consider the timestamp of check-ins. The Geographic-temporal-semantic-based location prediction model (GTS-LP) proposed by Ying *et al.*, extracts latitude and longitude information along with the time from every check-ins to predict the probable next location of visit for a user in LBSN [14].

Mobile phone call records are used in [31–33] for analysing human mobility patterns. Lu *et al.* [33] perform next location prediction for each user by analysing their uncertainties of movements, frequencies and temporal correlations of trajectories. For each active user they predict the next location which could be visited

on a specific day by utilising their historical data. A series of Markov chain models of order (1 to 7) is implemented for this purpose. The transition matrix and the frequency of visit at locations by the active users are repeatedly updated for every iteration in the Markov chain models. Finally, for each day the locations having high probability of transition from current location to the next location are predicted by their approach. Entropy in this work is used to measure the uncertainty or disorder of the trajectories. The entropy value is greatly reduced if both the spatial and temporal correlation in users' visit sequences are considered. From the obtained random entropy value, they stated that an individual can visit next any one out of four predicted locations. Moreover, if the frequency and sequence order of a trajectory are considered then this can be reduced to any one location out of two predicted locations.

Predicting hidden or unchecked locations from a published trajectory was first studied by Huo *et al.* [15]. They propose two inference modules; the hidden location finder and the probability estimator which finds the probability of an active user to visit a location. For any two consecutive check-ins l_i and l_{i+1}, the hidden location finder identifies a set of probable hidden locations between them. An A* algorithm is used to compute the shortest path between two consecutive check-ins. The Point of Interests (POI) falling on the shortest path are considered as the hidden locations. The probability estimator module estimates the Hidden Location Leakage Probability (HLPL) of an active user to visit each of the selected hidden locations. They use three different inference models (Baseline inference, Collaborative filtering and Hidden Markov Model) to compute the HLPL of the selected locations. The probable locations are ranked in descending order of their HLPL. Finally, the top-N locations from the ranked list is selected as the hidden locations between a consecutive check-in pair. These locations are treated as private and are provided to the active users as a privacy alert. However, our proposed work focuses only on identifying the hidden locations between two consecutive check-in location pairs. Throughout the paper we mention the work in [15] as HLPL. It can be noted here that our work does not focus in providing privacy alerts to the users.

3 Motivation and Problem Statement

As discussed in the previous section, the sequential check-ins and movement of similar users can be an effective source for location prediction. We consider a collection of these factors to deal with the problem of hidden location prediction. In this section, first we mention the major limitations of HLPL and also state how they are addressed in this work. Subsequently, we discuss the problem statement in details.

1. Associations between the checked-in locations are not considered for prediction. In the present work, we explicitly use the association rule mining technique to infer relationship between the published check-ins.
2. Time taken between two checked-in locations is computed only from check-ins of existing users. For example, say an active user follows a sequence $l_a \rightarrow l_b$, and there exists no other user who followed the similar sequence. In this case, HLPL fails to predict any hidden location between l_a and l_b. We address this problem by calculating the standard travel time of each user from the dataset.

3. The mobility of neighbours (similar users) is not explored for location prediction. It has been observed that neighbours or similar users often influence a user's mobility in a social network. In this approach, we first propose a novel measure to identify the similar users and then analyse their movements for location prediction.

We emphasize on the identification of hidden locations between consecutive check-in pairs in a published trajectory of an active user. The problem can be formally stated as follows.
Let $\mathcal{D} = \{S_1, S_2, S_3, ..., S_n\}$ be a check-in dataset of n users, where S_i is a check-in sequence of an active user i. A check-in sequence S_i consists of location information and check-in time at that location as shown below.

$$S_i = \langle (l_1, t_1), (l_2, t_2), ..., (l_m, t_m) \rangle$$

To keep our discussion simple, temporal information is omitted and the checked-in locations are arranged within a sequence in chronological order of their occurrence. Therefore, the published check-ins of the active user i can thus be represented as $S_i = \langle l_1 \rightarrow l_2 \rightarrow l_3 \rightarrow l_4 \rightarrow \ldots \rightarrow l_m \rangle$. It may be noted that a location can appear more than once in a sequence. The task of hidden location prediction is performed in three steps in the proposed work.

1. Select check-in pairs among $m-1$ consecutive check-ins as the potential candidates for hidden location prediction.
2. Predict a set of hidden or unchecked locations for each selected consecutive check-in pairs.
3. Rank the predicted locations in order of their chances of occurrence.

A few interrelated problems emerge from this. Can all the check-in pairs be considered for prediction? How many such locations should be predicted for each pair of consecutive check-ins? How do we rank those predictions? These questions are addressed in the following sections.

4 The Proposed Model

Let $S_u = \langle l_a \rightarrow l_b \rightarrow l_c \rightarrow l_d \rightarrow \ldots \rightarrow l_m \rangle$ be the published trajectory of an active user u. The hidden or unchecked locations between any two consecutive check-ins (say, $\langle l_c \rightarrow l_d \rangle$) are computed by exploring three important factors. First, the historical sequence of movement from l_a till the user reaches l_c is analysed. In the next step, proximal locations of l_c and l_d are considered. Finally, the next course of movement is considered from l_d till the end of trajectory l_m. The different modules of the proposed Associative Location Prediction Model (ALPM) as depicted in Fig. 1 are discussed next.

1. Check-in pairs are selected from published trajectory of users as the probable candidates for hidden location prediction. (*Consecutive Location Pair Selection*)
2. All pairs of associated locations are identified from trajectories of users in the social network by using an unsupervised learning technique. (*Infer Associated Locations*)
3. The Hidden Markov Model (HMM) and its standard inference algorithms are used to identify the hidden states or locations which generates a visible sequence of check-in. (*Hidden Markov Model*)

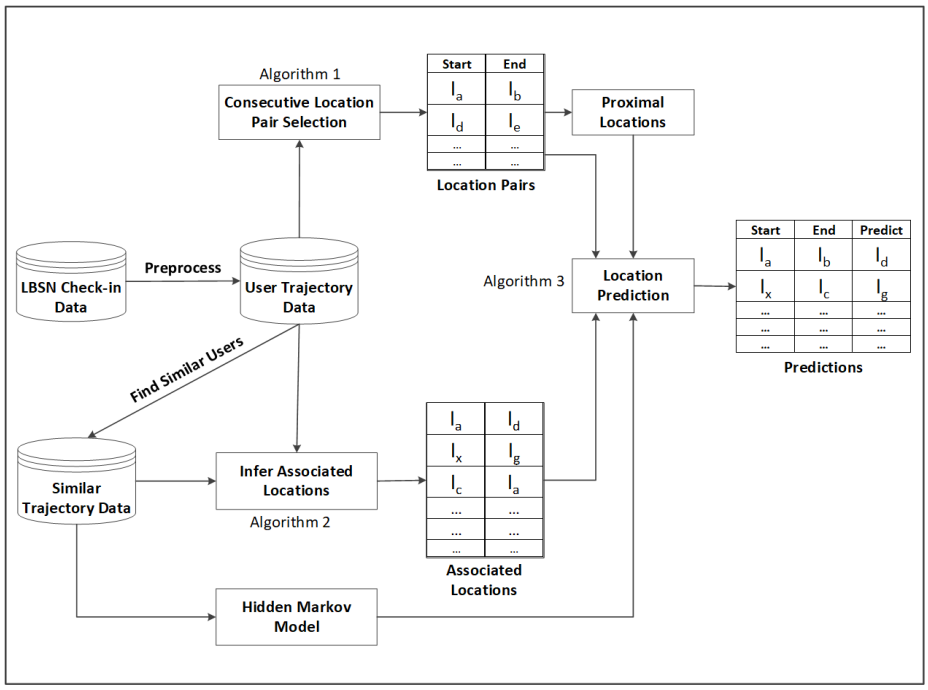

Fig. 1 Associative Location Prediction Model (ALPM)

4. The hidden or unchecked locations are predicted and ranked from the obtained associated locations, the unobserved locations from HMM and the geographically proximal locations. (*Location Prediction*)

Each of the modules is explained in detail in the subsequent subsections.

4.1 Consecutive Location Pair Selection

Let S_u be the check-in sequence of an active user u in a dataset $\mathcal{D}$. If the length of the sequence S_u is m, then there exist maximum of $m-1$ consecutive location pairs for hidden location prediction. Generally, hidden or unchecked locations appear within a few consecutive check-in pairs. Therefore, identifying those check-in pairs is considered as the first step for addressing the hidden location prediction problem. In this regard, an algorithm is devised to select only the potential candidates from the available $m-1$ pairs for hidden location prediction.

A check-in pair may be selected as a potential candidate for hidden location prediction if there exist users who have traversed them in less time. The average time T_{avg} taken to traverse the candidate location pair by the existing users is computed from their published trajectories. Hence, a scenario may occur when the candidate location pair is not checked-in by any existing user. In this case, the temporal stretch between the check-ins can never be found. However, we employ the geodesic time $T_{geodesic}$ which can still compute the temporal stretch between

the selected check-in pair. The $T_{geodesic}$ denotes the time taken to reach the next location from its immediate predecessor in the published trajectories. It is computed using the greater circle distance d_{gc} between the selected check-in pair and the average speed δ_{speed} of users in the social network. For computing δ_{speed}, we assume that a person checks-in to a place after he/she reaches it. Therefore, the average speed δ_{speed} for any user can be found from d_{gc} between each pair of its checked-in locations and the published time lag between them. Generally, check-ins by LBSN users vary across regions [4]. So we consider the trajectory of each individual user for computing the average speed. Therefore, our approach deals with two temporal variables T_{avg} and $T_{geodesic}$. The T_{avg} is the average time taken by other users to traverse a selected check-in pair. Whereas, $T_{geodesic}$ is computed using the distance between a selected check-in pair and the average speed δ_{speed} of every user in the social network. Henceforth, we select the maximum time between T_{avg} and $T_{geodesic}$, and denote it as T_{max}. If the time taken by a user u to traverse any consecutive location pair in S_u, is more than T_{max} then, that location pair is selected by our algorithm as a suitable candidate for further analysis.

Suppose, the check-in sequence S_u be $\langle l_a \rightarrow l_b \rightarrow l_c \rightarrow l_d \rangle$ for an active user u. For each of the three consecutive check-in pairs in the set, $\{l_a \rightarrow l_b, l_b \rightarrow l_c, l_c \rightarrow l_d\}$, we compute the geographical distance using the Haversine formula [16]. The standard distance measure such as the euclidean distance between two points is the straight line connecting them. However, the geographical distance between two locations on earth cannot be measured by a straight line connecting them. Hence, we consider the greater circle distance to find the distance between two locations along the surface. The straight lines connecting two locations are converted to a geodesic or curved lines over the sphere. The Haversine formula uses the latitude/longitude information and radius of earth to compute the greater circle distance d_{gc} between any two points on earth. We also arrange the selected consecutive sub-sequences or location pairs in order of their popularity among users. In this context, the term popularity refers to the number of times the consecutive location pair has been visited. Depicting this in detail we present Algorithm 1 for consecutive location pair selection. This algorithm neither depends upon whether a location pair is successively checked-in by any user, nor on the fact that all users are hiding a particular location within a published sequence. Next, we discuss how the association between locations are identified from published trajectories.

4.2 Frequent Location Set and Inference Rule Generation

In this proposed approach, association rule mining [17,18] is explored to infer the association between locations from published check-in trajectories of social network users. To the best of our knowledge, the association between a pair of check-ins has not been used for hidden location prediction. Though the association rule mining is primarily used over market basket data for predicting user activities [34], we argue that it can also provide significant information on user mobility from published trajectory data. Applying association rules over check-in data can effectively reflect correlation between check-ins.

Algorithm 1: Consecutive location pair selection for user u.

Input: $\mathcal{D}$: Social network dataset
Result: SS_u: Consecutive location pairs for user u

δ_{speed} is computed from distance and time lag between published check-ins;
for *each location i in trajectory of u* **do**
  $d_{gc} = haversine(l_i,\ l_{i+1})$;
  $T_{geodesic} = {}^{d_{gc}}/_{\delta_{speed}}$;
  for *each user j in $\mathcal{D}$ except u* **do**
    if *user j has checked-in at l_i and l_{i+1} consecutively* **then**
      $T_{SS_{STU}}[\,] = t_{l_{i+1}} - t_{l_i}$;
    end
  end
  $T_{avg} = average(T_{SS_{STU}})$;
  $T_{max} = maximum(T_{geodesic}, T_{avg})$;
  $T_u = t_{l_{i+1}} - t_{l_i}$;
  if $T_u \geq T_{max}$ **then**
    $SS_u = SS_u \cup \{(l_i, l_{i+1})\}$;
  end
end
return SS_u

Table 1 Sample check-in dataset.

UserID	LocationID
u1	a → b → c → d → e → f → g → h
u2	a → f → g
u3	b → d → e → f → b → d → j
u4	a → b → d → i → k
u5	a → b → e → g → e

The popular FP-growth algorithm [18] is deployed for mining frequent patterns of check-in locations in the proposed model. FP-growth can reduce the number of candidate location sets and the total number of database scans required. In the same line of FP-tree, a Frequent Location Pattern tree (FLP-tree) is generated. This prefix-like tree structure contains separate nodes for all checked-in locations having count more than the support threshold. In addition to it, each node also holds the number of times a location occurred in published trajectories. Every edge in the FLP-tree represents each unique location pair or the pattern of check-ins. We simplify this by stating an example of converting a sample check-in dataset into the FLP-tree. Consider a LBSN check-in dataset in Table 1 with 5 users and their sequence of check-ins over 11 locations. The generated FLP-tree shown in Fig. 2, consists of solid arrows depicting check-in sequences in published trajectories. Locations having support less than 2 are omitted from the check-in sequences. Dotted arrows connect and represent the locations involved in separate trajectories.

Next, the conditional pattern base for each location is computed. It is the collection of all possible patterns which ends with the checked-in locations. Table 2 shows the group of patterns forming the conditional pattern base for each location visited in the considered example. From this set of patterns, the associ-

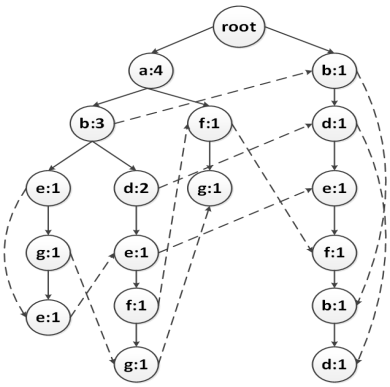

Fig. 2 FLP-tree with support threshold 2 from the published trajectories in Table 1.

Table 2 Generating frequent patterns or associate locations from the Conditional Pattern Base.

LID	Conditional Pattern Base	Conditional FLP-tree	Frequent Patterns
g	⟨a → b → e⟩, ⟨a → b → d → e → f⟩, ⟨a → f⟩	(a:3), (b:2), (e:2), (d:1), (f:2)	{a, b, e, f}|g
f	⟨a → b → d → e⟩, ⟨a⟩, ⟨b → d → e⟩	(a:2), (b:2), (d:2), (e:2)	{a, b, d, e}|f
e	⟨a → b⟩, ⟨a → b → e → g⟩, ⟨a → b → d⟩, ⟨b → d⟩	(a:3), (b:4), (e:1), (g:1), (d:2)	{a, b, d}|e
d	⟨a → b⟩, ⟨b⟩, ⟨b → d → e → f → b⟩	(a:1), (b:4), (d:1), (e:1), (f:1)	{b}|d
b	⟨a⟩, ⟨∅⟩, ⟨b → d → e → f⟩	(a:1), (b:1), (d:1), (e:1), (f:1)	{∅}|b
a	⟨∅⟩	(∅)	{∅}|a

ated locations or frequent patterns are generated along with their count in the generated patterns. The conditional FLP-tree thus formed is depicted in Table 2. From the conditional FLP-tree, the locations having count more than the support is selected for further analysis. Like, the associated locations for g and e are (a,b,e,f) and (a,b,d), respectively. Similarly, we find the associated locations for other check-ins as well. Generally, the support of a location to be considered as frequent 1-location set, depends upon the number of times it occurred in all the trajectories. It can be noted that if only similar trajectories are considered for performing association rule mining, then the number of frequent patterns generated will be large and relevant. Hence, along with performing association for all trajectories, we also perform association on trajectories of only the top-K similar users of an active user. Identifying the similar users or neighbours has been a very popular area of research in recent years [10,11,19].

In a LBSN, it is observed that a user's willingness to check-in at a location is often influenced by the published trajectories of friends and other similar users. Thus, the associated locations that are obtained from the trajectories of similar users can lead to more precise locations. Therefore, identification of those similar users is an important step in the location prediction problem. Similar users may be identified either by modeling a hierarchical location category tree [11] or by measuring the similarity between them based on their checked-in location histories.

Algorithm 2: Inferring associated locations.

Input: $\mathcal{T}$: Trajectories
Result: SFP: Selected frequent location patterns
FL = frequent locations from $\mathcal{T}$ having count more than support threshold
$FLPTree$ = construct FP-tree using locations in FL
CPB = construct conditional pattern base from $FLPTree$ for each location
CFT = conditional FP-Tree for each pattern in CPB
repeat
    for each CFT
    SFP = locations on single path in CFT
until resulting CFT is empty or single path
return SFP

Many standard similarity measures including Euclidean distance, Simple matching coefficient, Jaccard coefficient, Cosine similarity, Pearson's correlation coefficient, etc. may be used to identify the similar users [19]. The Jaccard coefficient in particular, has been found to be performing well as it considers the common locations as well as the number of different locations visited by users. However, in a social network the frequently checked-in locations drive interestingness and popularity of a particular location. Incidentally, the cosine similarity measure considers the frequency of occurrence of the check-ins in a trajectory. Therefore, we combine the Jaccard and Cosine similarity measure and term it the $JaCos$ measure. $JaCos$ considers both the common locations visited and the frequency of their visit for a pair of users. Let u and v be two random users in LBSN. If Cosine and Jaccard similarity measure are represented as $Cos(u, v)$ and $Jacc(u, v)$, respectively. Then $JaCos$ can be computed as:

$$JaCos(u, v) = 2 * \frac{Cos(u, v) * Jacc(u, v)}{Cos(u, v) + Jacc(u, v)} \tag{1}$$

Finally, top-K similar users of u are chosen and their trajectories for obtaining the associated locations are taken. Details of the FLP-tree generation and frequent pattern selection is depicted in Algorithm 2. It is applied twice on the dataset. Once with the trajectories of all users in the social network, and then with the trajectories of only the similar users of an active user. From a sequential data, the hidden states can be identified using various techniques such as Hidden Markov Model (HMM). Details of inferring hidden locations using the HMM is provided in the next section.

4.3 Inferring Hidden Locations Using HMM

Check-ins performed by users are generally driven by their interests, varying social trends, sentiments, etc. Predicting the mobility of a user on the basis of these components is a challenging task. Therefore, certain sophisticated techniques are required that can effectively identify mobility based on some hidden aspects. The Hidden Markov Model (HMM) and its variants have long been used to analyse the unobserved or hidden states in sequential data [20]. A standard representation of HMM is shown in Fig. 3. A HMM consists of some visible sequence of locations whose check-ins are influenced by some states which are predominantly hidden.

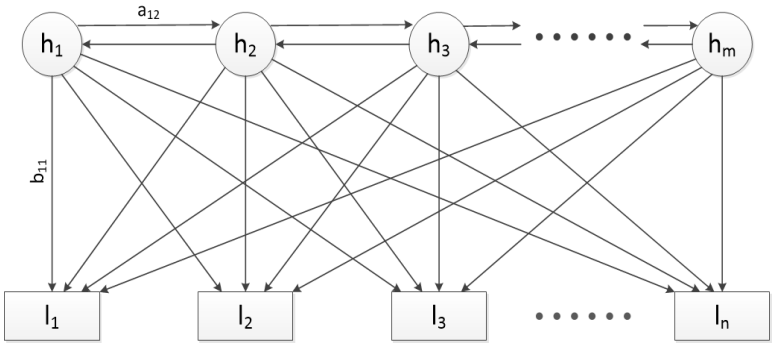

Fig. 3 A Hidden Markov Model with set of hidden states h and set of observed checked-in locations l.

These hidden states may be interpreted as the factors on which user mobility is influenced. The visible observations or the checked-in locations help to compute the probability of occurrence for every hidden states.

In Fig. 3, the observed check-ins are represented as $L = \{l_1, l_2, l_3, \ldots, l_n\}$ in square blocks. Each of these sequential checked-in locations are influenced by some unobserved or hidden factors which are represented as $H = \{h_1, h_2, h_3, \ldots, h_m\}$ in circles. The probability with which the model moves between the hidden states is termed as the transition probability of a hidden state. The probability of emitting an observed location from a hidden state is the emission probability. The transition probability and emission probability is represented by a and b, respectively as follows.

$$a_{ij} = P(h_j(t) \,|h_i(t-1)\,) \tag{2}$$

$$b_{jk} = P(l_k \,|h_j) \tag{3}$$

Here, the transition a_{ij} is the probability of moving from any hidden location h_i at $(t-1)$ to another hidden location h_j at time t. The emission b_{jk} is the probability to move from the hidden location h_j to a visible state or location l_k. It should be noted here that the final goal is to identify the distribution of hidden locations over the observed locations. From the conditional probabilities represented with arrows, it is clear that the probability of occurrence for a hidden location at time t depends only on the hidden location at $(t-1)$. The hidden location at $(t-2)$ or before has no effect on its distribution. As our problem also satisfies this property, the Hidden Markov Model can effectively be used to solve the problem of hidden location prediction. The model at any time step t, from a hidden state emits an observed location according to the emission distribution, and then moves to another hidden state at $(t+1)$ according to the transition probability. This property can be evaluated to solve three different problems, namely the evaluation problem, the decoding problem and the learning problem.

First, we formally define a Hidden Markov Model θ as (H, L, a_{ij}, b_{jk}) where, H is the set of all the hidden or unobserved factors on which the occurrence of a checked-in location depends. The set of observed checked-in locations obtained

from a user trajectory is represented as L. In this context, we include ν^T as the sequence of check-ins of length T visited by a user. An evaluation problem helps to identify the probability of generating a visible sequence of check-ins ν^T by a given Hidden Markov Model θ. This is achieved using a technique known as the *forward algorithm*. It can be stated as:

$$P(\nu^T \,|\theta) \;=\; \sum_{r=1}^{r_{\max}} P(\nu^T \,\Big|h_r^T\,) * P(h_r^T) \tag{4}$$

where, h_r^T is a sequence of hidden states within the collection of all hidden state sequences of length T. The maximum number of hidden sequences in the HMM is $r_{\max}$ and $P(h_r^T)$ is the probability of occurrence for a hidden state sequence. The term $P(h_r^T)$ can be computed as,

$$P(h_r^T) \;=\; \prod_{t=1}^{T} P(h_r(t)\,|h_r(t-1)) \tag{5}$$

Hence, using Equation (5) we can reformulate Equation (4) as,

$$P(\nu^T \,|\theta) \;=\; \sum_{r=1}^{r_{\max}} \left[\prod_{t=1}^{T} [P(\nu(t)\,|h_r(t)) * P(h_r(t)\,|h_r(t-1))\,] \right] \tag{6}$$

where, $\nu(t)$ is the emitted visible check-in at time t. While computing $P(\nu^T \,|\theta)$ using the forward algorithm, a matrix containing the posterior hidden state probabilities α is generated. It is the conditional probability $\alpha_j(t)$ of being at a hidden location h_i at time t after generating the first t number of visible check-ins given in ν^T. It can be formally stated as,

$$\alpha_j(t) = (b_{jk})_{\nu(t)} * \left[\sum_{i \in H} \alpha_i(t-1) * a_{ij} \right] \tag{7}$$

The hidden states at each time step having the highest probability can thus be explored to obtain the most probable sequence of unobserved locations through which the user has transited while generating the visible sequence of check-in ν^T. This is the decoding problem and is solved using the Viterbi algorithm [21].

The learning problem or training of the model, estimates the best possible transition a_{ij} and emission b_{jk} probabilities which maximizes the term $P(\nu^T \,|\theta)$ from Equation (7) and (8). Next, the estimated transition probabilities $\hat{a}_{ij}$ and emission probabilities $\hat{b}_{jk}$ are computed. This can be estimated from a special case of Expectation-Maximization (EM) algorithm called *forward-backward algorithm* or the Baum-Welch algorithm [22]. The forward algorithm produces the conditional probabilities for each hidden states in H, at each time step t and is thus represented as α (Equation (7)). On the contrary the backward algorithm $\beta_i(t)$ identifies the probability that the model θ will be in a hidden state h_i at time t and will generate the remaining sequence of ν^T starting from time $(t+1)$ to T. Formally, it can be represented as,

$$\beta_i(t) = \sum_{j \in H} \beta_j(t+1) * a_{ij} * (b_{jk})_{\nu(t+1)} \tag{8}$$

For the learning problem we define $\gamma_{ij}(t)$ as the probability of transition from h_i at time step $(t-1)$ to h_j at time step t for any particular visible sequence ν^T. It can be represented as,

$$\gamma_{ij}(t) = \frac{\alpha_i(t-1) * a_{ij} * b_{jk} * \beta_j(t)}{P(\nu^T|\theta)} \tag{9}$$

The denominator in Equation (9) finds the probability with which the model θ has generated the visible check-in sequence ν^T following any path (Equation (6)). Whereas, the numerator includes only that part where a transition from h_i to h_j from time step $(t-1)$ to t is involved (Equation (2), (3), (7) and (8)). This estimated $\gamma_{ij}(t)$ can henceforth be used to learn the improved values of a_{ij} and b_{jk}. From Equation (9), two issues can be identified which leads to the final estimation of the learning parameters. First, the expected number of transitions from $h_i(t-1)$ to $h_j(t)$ at any given time in the check-in sequence ν^T can be computed from $\sum_{t=1}^{T} \gamma_{ij}(t)$. Secondly, the total number of expected transitions from a hidden state h_i to any state can be found from $\sum_{t=1}^{T} \sum_{k \in H} \gamma_{ik}(t)$. The refined transition $\hat{a}_{ij}$ and emission $\hat{b}_{jk}$ probabilities can finally be estimated as,

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_{ij}(t)}{\sum_{t=1}^{T} \sum_{k \in H} \gamma_{ik}(t)} \tag{10}$$

$$\hat{b}_{jk} = \frac{\left[\sum_{t=1}^{T} \sum_{f \in H} \gamma_{jf}(t)\right]_{\nu(t)=\nu_k}}{\left[\sum_{t=1}^{T} \sum_{f \in H} \gamma_{jf}(t)\right]_{\forall \nu(t)}} \tag{11}$$

Initially, arbitrary values of a_{ij} and b_{jk} are chosen. Using those arbitrary values we estimate what will be the $\alpha_j(t)$ and $\beta_i(t)$ using Equation (7) and (8), respectively. The initial values of a_{ij}, b_{jk}, $\alpha_j(t)$ and $\beta_i(t)$ are used in Equation (9) to estimate the probability of transition $\gamma_{ij}(t)$. Further, it is used to estimate the refined values of $\hat{a}_{ij}$ and $\hat{b}_{jk}$.

The problem stated in this paper is addressed by first learning the refined emission and transition probabilities of the model using the Baum-Welch algorithm. Then the decoding problem or the Viterbi algorithm is used with the trained Hidden Markov Model and a visible sequence. Suppose, any user u_1 has a series of check-in locations $L = \langle l_1, l_2, l_3, l_4, l_5, l_6, l_7, l_8, l_9, l_{10} \rangle$. Let the Algorithm 1 (Section 4.1) be executed over L selects the consecutive check-in pair $\langle l_4, l_5 \rangle$ as the candidate consecutive locations having hidden locations. The Hidden Markov Model is used to identify the hidden or unobserved locations where u_1 is expected to visit before l_5 and after l_4. At first, the proposed *JaCos* similarity metric is used to identify the nearest neighbours of u_1. Subsequently, an arbitrary HMM is generated from the

Algorithm 3: Location prediction.

Input: SS_u: Consecutive location pairs for user u
SFP: Selected frequent location patterns
$HMMP$: Location probability matrix using HMM
$PROX$: Proximal locations of each location in SS_u
Result: $PREDICT$: Predicted hidden locations

```
for each location pair (l_p, l_q) ∈ SS_u do
    if a rule l_p → l_s exists in SFP then
        PREDICT = PREDICT ∪ {l_s} ;
    end
end
for each location pair (l_p, l_q) ∈ SS_u do
    PREDICT = PREDICT ∪ {location obtained from HMMP} ;
    PREDICT = PREDICT ∪ {proximal locations of l_p and l_q from PROX} ;
end
rank top-N locations in PREDICT using Entropy_Distance
return PREDICT
```

collection of check-ins of the similar users and also the series of check-ins performed by u_1 till l_5. From this arbitrary model, we generate the transition a_{ij} between unobserved locations and their probabilities to emit observed check-ins b_{jk}, using Equation (2) and (3), respectively. To learn the refined values of a_{ij} and b_{jk}, we train the model using the visible check-ins ν performed by u_1 from l_4 to l_{10}. We continue refining the transition $\hat{a}_{ij}$ and emission $\hat{b}_{jk}$ probabilities using Equation (10) and (11), respectively till they change in successive iterations. Finally, the Viterbi algorithm is used with the refined transition and emission probabilities. As mentioned earlier, it is used to compute the conditional probability that the user has visited a hidden location before moving to l_5 from l_4. The hidden location having highest probability is thus predicted as the unobserved location between the selected check-in pair $\langle l_4, l_5 \rangle$.

4.4 Predicting Unchecked Locations

In this section, we briefly describe how the consecutive location pair selection method (Section 4.1), the FP-growth association rule mining technique (Section 4.2) and the Hidden Markov Model (Section 4.3) are combined for predicting unchecked locations. Let SS_u be the selected location pairs for a user u in a dataset $\mathcal{D}$ and the selected frequent location patterns obtained from the FP-growth algorithm be SFP. Consider SS_u as $\{l_a \rightarrow l_b, l_c \rightarrow l_d\}$ and a portion of SFP as $\{(l_a, l_p), (l_q, l_b), (l_a, l_s), (l_c, l_w)\}$. Our proposed unchecked location prediction method checks whether the start locations of the selected pairs (like, l_a and l_c) exist as antecedents in each pair within SFP. If any such location rule is found in SFP then the corresponding consequent location is extracted. The selected location forms the predicted unchecked location between the first and second location of a check-in pair in SS_u. In the considered case, the predicted location set PL will be (l_p and l_s) for check-in pair $\langle l_a, l_b \rangle$ and (l_w) for check-in pair $\langle l_c, l_d \rangle$. Along with it the locations obtained after employing HMM is also added to the prediction set.

The locations obtained from frequent check-in pairs are hereby considered as hidden or unchecked locations. However, there may exist some candidate check-in locations which are not involved in any frequent location set. In this case, the proposed model will fail to predict the unchecked locations. However, a user always has a very high chance of passing by the nearby places between two sequential check-ins. Taking this scenario into consideration, we explore the proximal locations as a candidate for the hidden location prediction set. For every candidate check-in pair obtained from Algorithm 1, we find the proximal locations and also include them in the set of predicted locations.

Incidentally, within a selected pair of subsequent check-ins there can be multiple unchecked locations. Our proposed method predicts multiple locations for each selected check-in pair and therefore it is necessary to rank the predicted locations. Various ranking techniques have been proposed in this regard. The standard ranking techniques are based on probabilistic measures like, Bayesian inference [15, 23], Collaborative filtering [24–26], etc. Bayesian inference finds the conditional probability of occurrence for an event, with a given prior belief that a series of events has already occurred. Collaborative filtering is a method of filtering or predicting various events by combining the preferences of other collaborating users in the social network. The above two standards have been widely used to measure the probability of occurrence for various events across LBSNs. However, for ranking predicted locations, popularity, diversity and proximity of locations should be considered in a collaborative way.

In this work, we introduce a ranking technique termed as Entropy_Distance. It can rank the top-N locations from the predicted location set PL for every selected location pair on the basis of their chances of occurrence. Entropy can be an important component to judge the effectiveness of a predicted location. Prediction of a location is always driven by its popularity among users. Entropy is the measure of how many different users have visited a location [12, 33]. It is computed by considering the number of users observed at a location along with the rate in which the users have visited the location. The distance of the predicted hidden location and the published check-in is also an important factor for ranking. Therefore, we incorporate both the entropy and distance component in the Entropy_Distance ranking technique. The objective here is to rank the most populated, diverse and nearest locations first in the predicted list. Entropy_Distance (ED) of any predicted hidden location HL for a selected consecutive check-in pair $\langle l_1, l_2 \rangle$ can be represented as,

$$ED_{HL} = \frac{-\sum_{x \in U} P(x) \log_2 P(x)}{Average\, of\, distance(HL, l_1)\, and\, distance(HL, l_2)} \tag{12}$$

where, $P(x)$ is the ratio of the number of times user x has checked-in at HL to the number of times all users have checked-in at HL and U is the number of users in the social network. $distance(HL, l_1)$ is the distance between locations HL and l_1. If the observed check-ins are in close proximity, then the denominator (average distance) of ED will differ little. In such a situation, the entropy of locations will drive their ED values. Entropy considers both the frequency of visits by users at a location and the number of unique users who visited the location. A non-popular

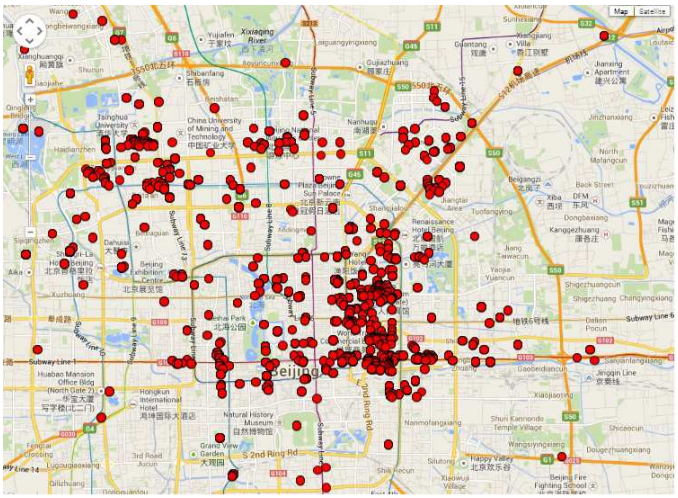

Fig. 4 Checked-in locations at Beijing.

location does not have large number of check-ins from a wide range of users. Thus for a non-popular location the entropy is found to be less than a popular location. Therefore, the Entropy_Distance of a popular location will be higher than a non-popular location. On the other hand, say two non-popular locations l_1 and l_2 exist, where l_1 is closer to both the consecutive check-in pair. The ED of l_1 will be higher than l_2 as the denominator in ED_{l_2} is larger than ED_{l_1}. Algorithm 3 depicts the entire approach of predicting and ranking the unchecked locations.

5 Experimental Results and Performance Analysis

We conducted a series of experiments over real-world Gowalla check-ins dataset [12]. All the experiments were executed on an Intel-i7 processor having 12GB of RAM and running on Microsoft Windows 8.1. We used the numerical computing environment of Matlab 2013a. This section starts with the description of Gowalla users' check-in dataset followed by the evaluation method, experimental results and analysis.

5.1 Data Preparation

Gowalla check-in data collected during Feb. 2009 to Oct. 2010 is used to perform experiments. It consists of 6,442,892 check-ins by 107,092 users with 1,280,969 unique locations checked-in worldwide. It consists of the following properties: 1) user identity; 2) checked-in time; 3) latitude and longitude information of each checked-in location; 4) location identity for each unique locations checked-in (Table 3).

The selected consecutive pairs of check-ins for hidden location prediction depend upon a time duration. Therefore, it is very likely that a hidden location between the consecutive pair of check-ins will reside within a feasible distance. To

Table 3 A sample of Gowalla check-in dataset.

uid	time	lat	long	locid
175	'2010-05-07T02:07:27Z'	40.0775460263000	116.586227417000	23254
175	'2010-05-07T02:08:51Z'	40.0753787039000	116.601505279500	733940
599	'2010-07-11T14:07:35Z'	40.0775460263000	116.586227417000	23254
599	'2010-08-13T21:00:13Z'	40.0775460263000	116.586227417000	1242911
599	'2010-08-17T05:30:31Z'	40.0775460263000	116.586227417000	788460
866	'2010-05-21T12:58:49Z'	39.8957944406000	116.463148833900	1147091
3449	'2010-08-19T02:32:29Z'	39.9854440000000	116.473312000000	1210328
3449	'2010-08-19T02:35:01Z'	39.9779231000000	116.473720333300	1056803

capture this notion, we experimented on check-ins within a confined geographical area (city). The Gowalla dataset is preprocessed by selecting only those check-ins performed within the geographical boundary of Beijing city, China. The selected check-ins are plotted over a real-life geographical map as shown in Fig. 4. It is observed that the frequency of check-ins by the users are sparsely distributed.

The check-in dataset is transformed into a sequence trajectory dataset (a sample is shown in Table 1), where each row includes all sequential check-ins performed by a user. Fig. 5(a) shows the frequency of check-ins performed by each user of the extracted dataset in Beijing. It is observed that the frequency of check-ins by the users are highly skewed. So in the experiments only those users who have checked-in at least five unique locations throughout their check-in history are selected. Similarly, Fig. 5(b) shows how frequently the locations were visited or checked-in by users. Less frequently visited locations will not add much knowledge to the trajectory analysis purpose. Therefore, the dataset is further reduced by considering only those locations which have been visited by at least two users at different timestamps. Fig. 6(a) and Fig. 6(b) shows the frequency of check-ins for each user in the social network and the number of times the locations are visited after reduction, respectively. Detail description of the Beijing dataset after preprocessing is given in Table 4.

For experimentation, we deliberately hide 95%, 90%, 85%, 75%, 50% and 25% random locations from the published trajectory of each user. In this way, we generate six different training sets having 5%, 10%, 15%, 25%, 50% and 75% of published check-ins to evaluate and compare the proposed model (ALPM) with the existing technique (HLPL [15]). These six check-in datasets are successively placed as input to the models and their performance to predict the deliberately hidden locations is evaluated. This evaluation technique helps to analyse the performance of ALPM with both sparse and dense data. The training dataset having 5-25% published check-ins for each user is considered as sparse, whereas 50-75% of published check-ins for each user is considered as dense. As an output of the ALPM, we collected top-5, 10 and 15 predicted hidden locations for each deleted check-in in the training dataset. Finally, the performance of ALPM is evaluated using standard metrics. The deleted locations in each dataset are henceforth used as the ground truth for performance evaluation.

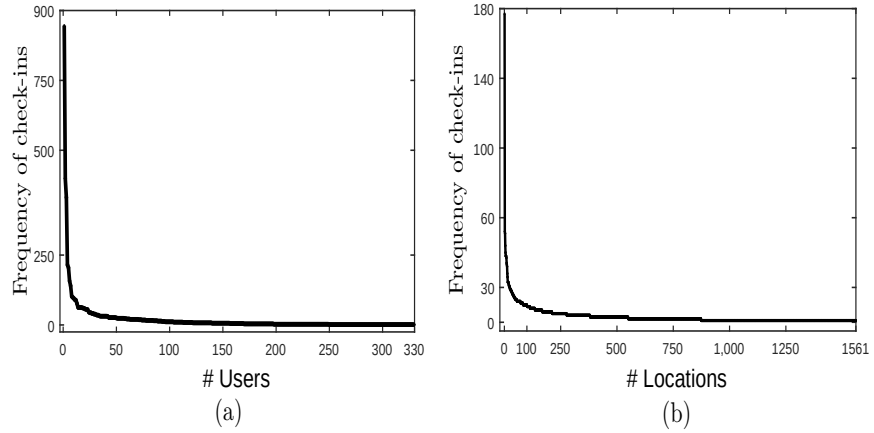

Fig. 5 Check-in statistics of Beijing city before reduction. (a) Frequency of check-ins by each user. (b) Frequency of check-ins at each location.

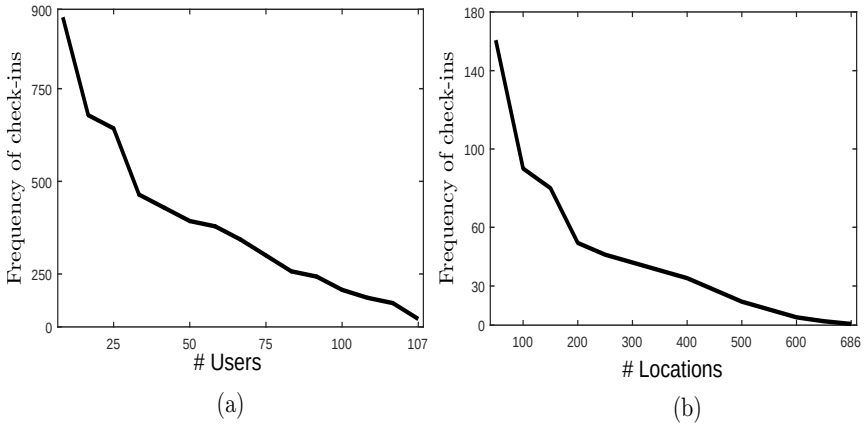

Fig. 6 Check-in statistics of Beijing city after reduction. (a) Frequency of check-ins by each user. (b) Frequency of check-ins at each location.

5.2 Evaluation Metrics

The proposed ALPM is executed in five different phases to justify importance of each module as described in Section 4. To show the importance of the first module, we evaluate ALPM with and without it. Results marked as ALPM-AS were obtained by experimenting without the consecutive location pair selection module (Section 4.1). The rest of the experimentation uses this module for selecting consecutive location pairs from user trajectory. This gives us a clear idea, why selecting

Table 4 Detail description of working dataset.

Total Check-ins	**Total users**	**Area Size**(km^2)	**Check-in interval**(h)
3655	107	16807.8	65.0495
Density($user/km^2$)	**Check-in/user**	**Check-in/POI**	**Check-in distance**(km)
0.02	15.83	3.3494	5.5698

the potential check-in pairs is essential for the hidden location prediction problem. ALPM-FP uses only the frequent location set module (Section 4.2) for predicting the hidden locations from a user's trajectory. It should be noted that the HMM and the proximal locations are not used in ALPM-FP. Further, ALPM-HMM represents the results obtained by considering only the Hidden Markov Model to predict the hidden locations. Finally, the advantage of ranking the predictions is evaluated by experimenting with and without the proposed Entropy_Distance ranking technique. ALPM and ALPM-UR are the results obtained with and without using the ranking technique, respectively. Here we mention the parameter setting for evaluating our proposed approach. For the current work, the support threshold is set at 0.2 for inferring the associated locations (Algorithm 2). The top-20 similar users are selected for each active user. The locations within 2Km of the selected consecutive check-in pairs are considered as the proximal locations.

For performance evaluation we compute the following measures; precision, recall and F1-Measure. For the sake of readability, we present them below.

Precision or Positive Predictive Value is the fraction of retrieved locations that are relevant. It can be represented as the ratio between the correct predictions to the total number of predictions.

$$Precision = \frac{\#\,Recovered\ ground\ truths}{\#\,Predictions} \tag{13}$$

Recall or Hit Rate is the fraction of relevant locations that are retrieved. It can be represented as the ratio between the correct predictions to the total number of correct locations.

$$Recall = \frac{\#\,Recovered\ ground\ truths}{\#\,Ground\ truths} \tag{14}$$

For any test, there is always a trade-off between the precision and recall. Therefore, $F1\text{-}Measure$ is frequently used in the literature.

F1-Measure or F-Score is the weighted average of the precision and recall.

$$F1\text{-}Measure = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{15}$$

5.3 Results and Analysis

Plots in Fig. 7, Fig. 8 and Fig. 9 show the performance evaluation metric values (in %) for each percentage of check-ins provided as input to the models. We compare

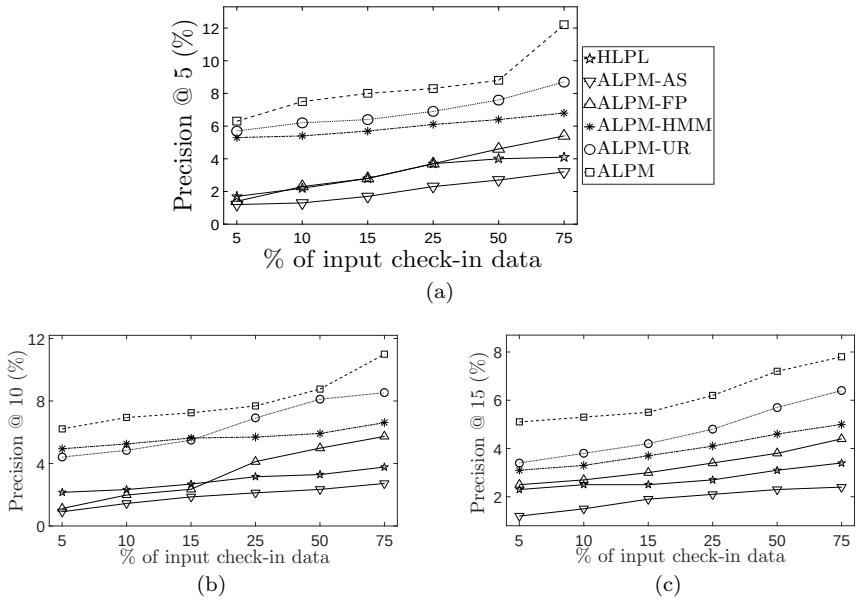

Fig. 7 Precision of our proposed approach and the state-of-art technique in [15]. (a) Precision of the top-5 predicted hidden locations vs the percentage of check-ins provided as input to the models. (b) Precision of the top-10 predicted hidden locations vs the percentage of check-ins provided as input to the models. (c) Precision of the top-15 predicted hidden locations vs the percentage of check-ins provided as input to the models.

the proposed model ALPM with the existing work in [15]. The percentage of published check-ins provided as input to the model is varied over the horizontal axis on all the plots. For evaluation purpose, we increase the percentage of check-ins provided to a model from 5% to 75%. For each hidden location (ground truth) we also increase the number of predictions N from 5 locations to 15 locations. The performance evaluation metric values are weighted by the total number of users in the dataset. The metric values are computed for the predicted locations by HLPL [15], ALPM-AS, ALPM-FP, ALPM-HMM, ALPM-UR and ALPM. The ALPM-AS uses all the check-in pairs for finding the hidden locations. The ALPM-FP and ALPM-HMM uses the frequent patterns and the Hidden Markov Model, respectively to identify the hidden locations. The ALPM-UR combines Algorithm 1, the frequent patterns, the Hidden Markov Model and the proximal locations to predict the hidden locations. However, predictions from ALPM-UR are not ranked. Finally, ALPM combines all the modules included in ALPM-UR and also ranks the predictions. Subsequent results show requirement of each of the modules mentioned in Fig. 1.

Fig. 7(a) shows that the precision of HLPL is higher than ALPM-AS. It can be noted that ALPM-AS does not use the Algorithm 1 for selecting the potential check-in pair with hidden locations. Instead, it considers all the check-in pairs as candidates for hidden location prediction. This leads to many irrelevant location predictions, and thus lowers the precision. The ALPM-FP uses the Algorithm 1 for

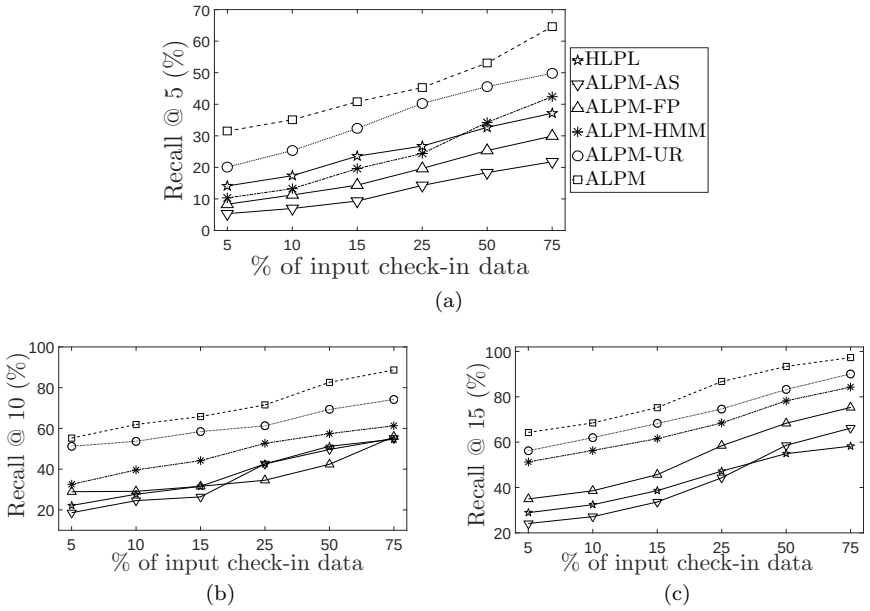

Fig. 8 Recall of our proposed approach and the state-of-art technique in [15]. (a) Recall of the top-5 predicted hidden locations vs the percentage of check-ins provided as input to the models. (b) Recall of the top-10 predicted hidden locations vs the percentage of check-ins provided as input to the models. (c) Recall of the top-15 predicted hidden locations vs the percentage of check-ins provided as input to the models.

selecting candidate check-in pairs for hidden location prediction unlike the HLPL. With top-5 predicted locations (N=5) we observe that for sparse data (5-25%) the precision of ALPM-FP is at par with HLPL. However, the precision obtained from ALPM-FP for dense data (50-75%) supersedes HLPL. The ALPM-FP only uses the association between locations. The results marked as ALPM-HMM uses only the predictions obtained from Hidden Markov Model and does not use the association between locations for location prediction. The ALPM-HMM outperforms the existing HLPL for any number of predicted locations. The combined approach ALPM-UR achieves better performance than the ALPM-HMM. Similar trends are observed over various percentage of check-ins provided as input to the model. For the ranked ALPM we observe best precision with top-5 predictions from 75% of check-in data (Fig. 7(a)). With top-5 predictions from 75% of check-in data, we obtain close to 12% of precision from ALPM compared to 4%, 3%, 5.5%, 6.8% and 8.7% from HLPL, ALPM-AS, APLM-FP, ALPM-HMM and ALPM-UR, respectively. Similar trends are observed with more number of predicted locations (N=10 and 15). For a dense dataset (50-75%) with more number of predicted locations, the precision obtained from ALPM-FP is found to outperform HLPL. We also observe that with top-10 predictions the ALPM-HMM is superior to the ALPM-UR for sparse dataset (Fig. 7(b)). However, better results are obtained for ALPM-UR with increase in percentage of input check-in data. The ranked predictions (ALPM) outperforms ALPM-UR for any percentage of check-in data. From Fig. 7(c) it is observed that all the individual models of ALPM (ALPM-

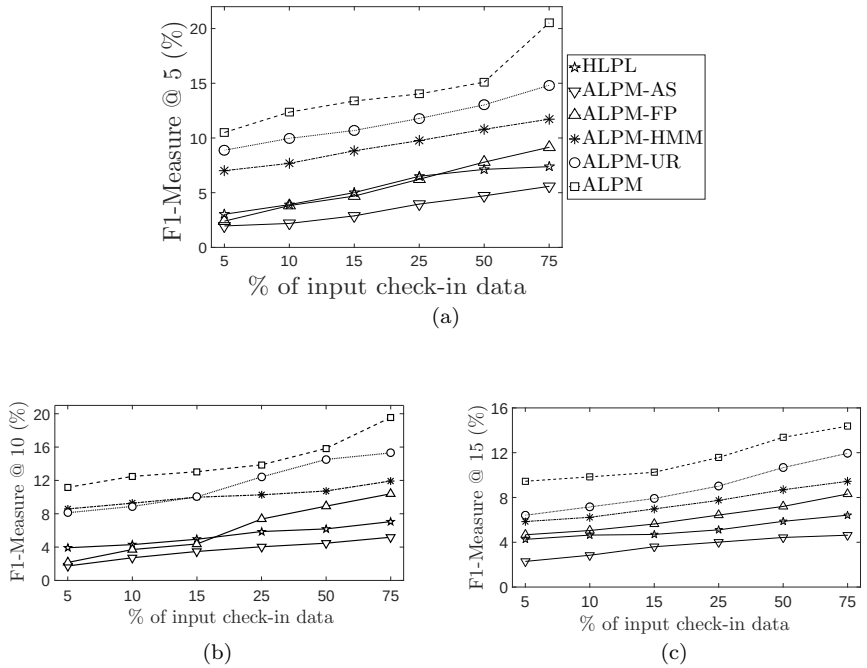

Fig. 9 $F1$-$Measure$ of our proposed approach and the state-of-art technique in [15]. (a) $F1$-$Measure$ of the top-5 predicted hidden locations vs the percentage of check-ins provided as input to the models. (b) $F1$-$Measure$ of the top-10 predicted hidden locations vs the percentage of check-ins provided as input to the models. (c) $F1$-$Measure$ of the top-15 predicted hidden locations vs the percentage of check-ins provided as input to the models.

FP, ALPM-HMM and ALPM-UR) including the proposed model ALPM obtained much improved results than the existing approach HLPL.

The HLPL outperforms ALPM-AS and ALPM-FP in recall measure with top-5 predictions. However, when all the modules mentioned in Fig. 1 are combined (ALPM), we observe a superior value for the recall measure. The predictions from Hidden Markov Model (ALPM-HMM) outperforms HLPL for dense dataset. With top-5 predictions from 75% of check-in data the obtained recall from ALPM-HMM is close to 42% compared to 37% with HLPL (Fig. 8(a)). The HMM based model ALPM-HMM is observed to achieve better result than ALPM-FP, ALPM-AS and HLPL for all variations of input check-in data. The combined approach ALPM-UR and ALPM retrieve more number of relevant locations, as observed from the accompanying results for top-10 predictions (Fig. 8(b)). ALPM obtains close to 89% of recall with top-10 prediction from 75% of check-in data as compared to 48% from HLPL. With top-15 predictions, the ALPM-AS model does not perform well with a sparse dataset, whereas it overcomes HLPL for a dense dataset. Plots in Fig. 8(c) clearly show that our proposed approach ALPM supersedes HLPL in the recall measure. The best recall of our approach is found to be 91.3% with 75% of check-in data and top-15 locations predicted.

The results obtained for F1-Measure are important due to the fact that it deals with the trade-off between precision and recall. From Fig. 9(a), it is observed that ALPM-AS does not perform well for less number of predictions (N=5). With top-5 predictions, the performance of ALPM-FP and the existing HLPL are almost at par for sparse dataset. Whereas for a dense dataset the ALPM-FP is found to be performing better than HLPL. However, the HMM based ALPM-HMM, the unranked predictions ALPM-UR and the ranked predictions ALPM supersedes the existing HLPL with top-5 predictions (Fig. 9(a)). Top-10 predictions obtained from ALPM-FP achieves better results than HLPL when 25-75% of check-ins are used. With top-10 predictions from 5% check-in data the F1-Measure is observed to be close to 2% and 4% for ALPM-FP and HLPL, respectively. Whereas, with top-10 predictions from 75% check-in data the F1-Measure is observed to be close to 10.4% and 7% for ALPM-FP and HLPL, respectively (Fig. 9(b)). ALPM is again found to achieve better performance than other models. With top-15 predictions the proposed ALPM is found to obtain superior results than the existing HLPL (Fig. 9(c)). For the current experimental setup, our proposed model ALPM achieves best result when top-15 locations are predicted using 75% of check-in data. Next, we test which technique is suitable for ranking the locations obtained using ALPM.

5.4 Ranking the Predictions

Generally, the methods in the literature predict a set of locations against a single hidden location. Therefore, there is a need for ranking the predictions. Bayes' law [23] and Collaborative filtering techniques [25, 26] can be used for ranking the predictions. Work proposed in [15] ranks the predicted locations using the mentioned standard metrics. For comparison the proposed approach also ranks the selected locations using the Bayes' law and two Collaborative filtering techniques. In addition, a new ranking metric termed as Entropy_Distance (ED) is also introduced. Next, we discuss each metric in detail.

Bayesian inference (BI) as depicted in [15, 23] is based on the Bayes' law and it computes the conditional probability with which a user visits a location B after reaching a location A (Equation (16)).

$$P(B|A) = \frac{P(A|B) * P(B)}{P(A)} \tag{16}$$

Collaborative filtering [25] states that if any user u_1 has followed the same sequence of movement as user u_2 for first c check-ins, then it is highly probable that the next sequence of movement, $(c+1)$ for user u_2 can be analysed by further forward movement pattern of user u_1. User based collaborative filtering technique (UCF) involves only the similar users in the network for analysis. Hence, as depicted in [15], if $U = \{u_1, u_2, u_3, \ldots, u_n\}$ is the set of users and $L = \{l_1, l_2, l_3, \ldots, l_q\}$ is the locations checked-in by them, then the probability of any user u_i to visit a location l_j is given by,

$$UCF_{u_i,l_j} = \frac{\sum\limits_{z \in U} (SI_{u_i,u_z} * T_{u_z,l_j})}{\sum\limits_{z \in U} SI_{u_i,u_z}} \tag{17}$$

where,
SI_{u_i,u_z} = cosine similarity measure between users u_i and u_z,
$T_{u_z,l_j} = \begin{cases} 1; \text{if } u_z \text{ has checked in at } l_j \\ 0; \text{if } u_z \text{ has not checked in at } l_j \end{cases}$

Friend based collaborative filtering (FCF) [26] differs from UCF by the fact that the similarity index SI is computed on basis of the social influence of friends or nearest neighbours. Here, social influence is the measure of how a user's social group influences his/her mobility across the online social network. During experimentation, to compute the social influence between a pair of users we combine the mutual friends and the locations checked-in by them. The social influence SI_{u_i,u_z} of a user u_i over any other user u_z is computed as,

$$SI_{u_i,u_z} = \sigma * \left| \frac{F_i \cap F_z}{F_i \cup F_z} \right| + (1-\sigma) \left| \frac{L_i \cap L_z}{L_i \cup L_z} \right| \tag{18}$$

where,
$F_i,\ F_z$ = friends of u_i and u_z, respectively,
$L_i,\ L_z$ = checked-in locations for u_i and u_z, respectively, and
σ = the turning factor in the range [0,1].

In practice, only the similar users may not be the ultimate source of quality knowledge and experience. However, the social friends often hold important information which may guide towards location prediction. Here, we have considered the turning factor as 0.9 to give more importance to the social ties. Next, we compare the above mentioned ranking metrics with the proposed Entropy_Distance metric (discussed in Section 4.4).

The performance of ranking techniques can be evaluated by observing the position of each hidden location in the top-N predicted list. In our case, it is desired that the ground truth should be ranked as the first location in the list (if only one location is hidden). If two ranking metric $G1$ and $G2$ rank the ground truth at positions i and j, respectively where $i < j$, then $G1$ is favoured to be a better performing ranking metric than $G2$. To capture this aspect, we use variant of a recently published ranking metric ***Positional Rank*** [30] for evaluation. Let R be set of all hidden locations (ground truth) for a user u_1. The *Positional Rank* of user u_1 is computed as:

$$Positional\ Rank_{u_1} = 1 - \frac{1}{|R|} \sum_{i \in R} \left[\frac{|PR_i - OR_i|}{|PL|}, if\ \{i \in R\} \cap PL \neq \emptyset;\ 0,\ otherwise \right] \tag{19}$$

where, PR_i is the predicted rank and OR_i is the original rank of the hidden location i in the predicted list PL. The *Positional Rank* of the retrieved hidden locations for all users U is the average of the *Positional Rank* of retrieved hidden locations for each user. It can be computed as:

$$Positional\,Rank = \frac{1}{|U|}\sum_{i \in U} Positional\,Rank_i \tag{20}$$

Positional Rank value ranges from 0 to 1, with a value near to 1 signifying a better ranking technique. In addition to it, we also evaluate the ranking technique using standard metrics like Mean Reciprocal Rank (MRR) and Discounted Cumulative Gain (DCG).

Mean Reciprocal Rank (MRR) is a commonly used evaluation metric in information retrieval [27,28]. For a given prediction list, it observes how early the relevant documents are ranked. In our case, let R be the set of all hidden locations (ground truth) for a user u_1. The MRR for user u_1 is the reciprocal of the predicted ranks of the relevant hidden locations, averaged over R. It can be represented as:

$$MRR_{u_1} = \frac{1}{|R|}\sum_{i \in R}\left[\frac{1}{r_i}, if\,\{i \in R\} \cap PL \neq \emptyset\,;\, 0,\, otherwise\right] \tag{21}$$

where, r_i is the predicted rank of the relevant hidden location i in the predicted list PL. The MRR of the retrieved hidden locations for all users U is the average of the MRR of retrieved hidden locations for each user. It can be computed as:

$$MRR = \frac{1}{|U|}\sum_{i \in U} MRR_i \tag{22}$$

Discounted Cumulative Gain (DCG) can also be used to evaluate a ranking technique [28,29]. It uses a graded relevance scale for each ranked elements, to compute its measure of usefulness in the predicted list. Like the MRR metric, it also assumes that a relevant element should appear before an irrelevant element in the predicted list. In our case, relevance of the ranked predicted hidden locations is binary, i.e. relevance is {0,1}. If the hidden location (ground truth) exists in the predicted list then its relevance is 1 else the relevance is 0. Therefore, the DCG for a user u_1 with R number of hidden locations (ground truth) can be reformulated as:

$$DCG_{u_1} = \frac{1}{|R|}\sum_{i \in R}\left[\frac{2^{relevance}-1}{\log_2(r_i+1)}, if\,\{i \in R\} \cap PL \neq \emptyset, relevance = 1; 0,\, otherwise\right] \tag{23}$$

where, r_i is the predicted rank of the relevant hidden location i in the predicted list PL. The DCG of the retrieved hidden locations for all users U is the average of the DCG of retrieved hidden locations for each user. It can be computed as:

$$DCG = \frac{1}{|U|}\sum_{i \in U} DCG_i \tag{24}$$

We compute the *Positional Rank*, MRR and DCG of all the retrieved hidden locations from ALPM by varying the percentage of check-ins available in trajectory of each user and also the number of locations predicted N from 5 to 15. Fig.

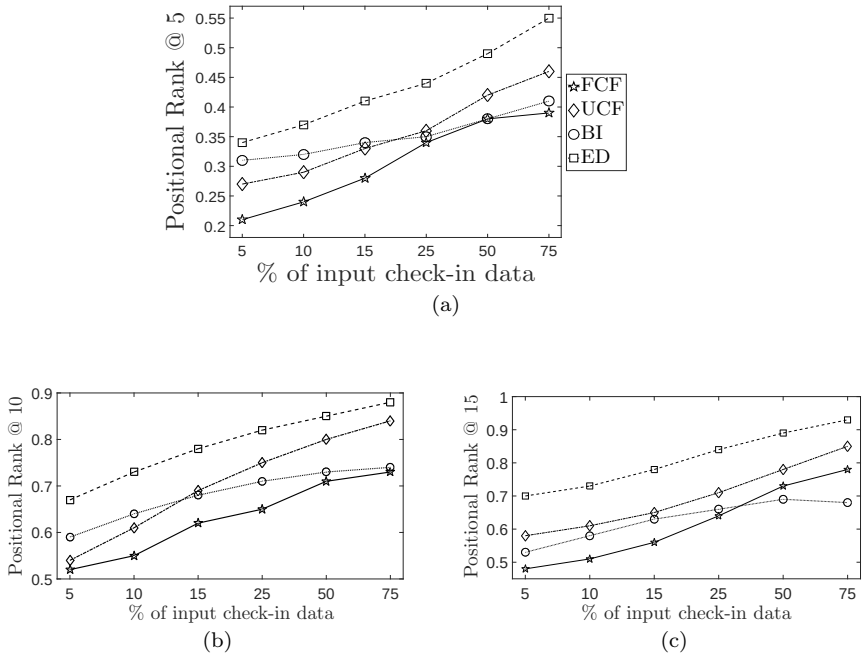

Fig. 10 *Positional Rank* of our proposed ranking technique and the existing techniques for ALPM. (a) *Positional Rank* of the top-5 predicted hidden locations vs the percentage of check-ins provided as input to ALPM. (b) *Positional Rank* of the top-10 predicted hidden locations vs the percentage of check-ins provided as input to ALPM. (c) *Positional Rank* of the top-15 predicted hidden locations vs the percentage of check-ins provided as input to ALPM.

10 shows the *Positional Rank* obtained by applying the proposed ranking metric Entropy_Distance and other standard metrics in ALPM. It is observed that the ranking using Bayesian inference has comparatively less effect on the number of check-ins input to the ALPM. Collaborative filtering techniques of user based and friend based mostly depend upon the nearest neighbours and the social ties, respectively. We observe better *Positional Rank* for both the methods, as the percentage of input check-in data (5 to 75%) and the number of predicted locations (5 to 15) increases. However, our proposed metric ED outperforms the existing ranking metrics. Best result for *Positional Rank* using ED is obtained when top-15 locations are predicted. The value of *Positional Rank* with top-15 predicted locations (N=15) at 5% and 75% of input check-in data is close to 70% and 93%, respectively compared to 58% and 85%, respectively for the UCF metric (Fig. 10(c)).

The results for MRR show that ranking the predictions from ALPM using BI produces better result than FCF and UCF with sparse data (Fig. 11). However, for the proposed model ALPM, the FCF ranking technique achieves better result than BI with top-15 locations predicted and having 75% of input check-in data. The MRR with top-15 locations having 75% of input check-in data, rises close to 36% compared to 34% for BI (Fig. 11(c)). In the same line, UCF outperforms BI for a dense dataset. This is inline with the fact that collaborative filtering depends

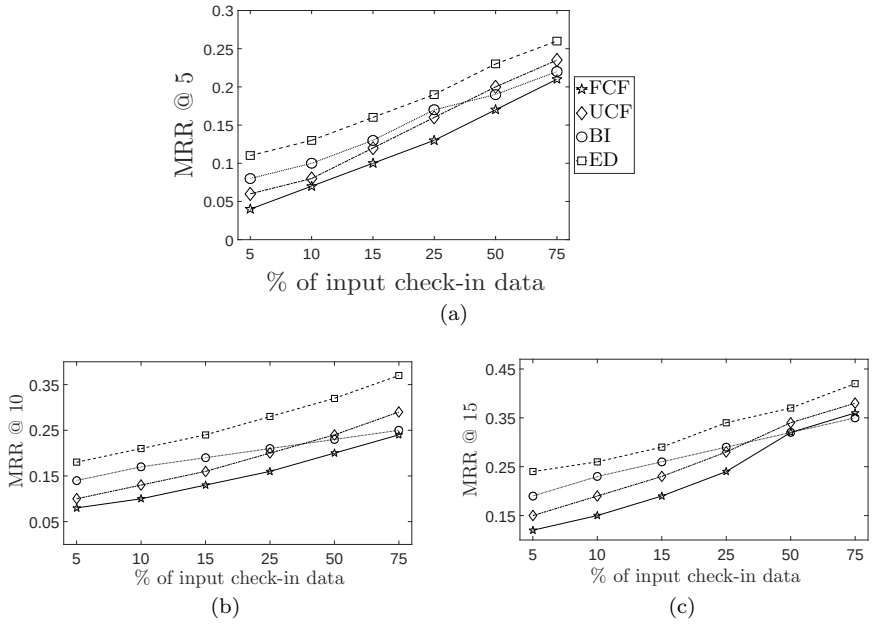

Fig. 11 MRR of our proposed ranking technique and the existing techniques for ALPM. (a) MRR of the top-5 predicted hidden locations vs the percentage of check-ins provided as input to ALPM. (b) MRR of the top-10 predicted hidden locations vs the percentage of check-ins provided as input to ALPM. (c) MRR of the top-15 predicted hidden locations vs the percentage of check-ins provided as input to ALPM.

upon activities of related users. Hence, with the increase of check-in data the performance of collaborative filtering also excels. However, the MRR results show that the proposed ED metric outperforms the collaborative filtering techniques and the Bayesian inference (Fig. 11).

Results with another evaluation metric DCG shows that the proposed ranking metric ED outperforms BI and the collaborative filtering techniques (Fig. 12). The ED achieves a higher DCG with any percentage of input check-in data when compared with the other ranking metrics. Among the existing metrics, only the UCF was found to obtain nearly comparable values with ED in DCG. With top-15 predicted locations from 75% of check-in data, ED obtains close to 74% in DCG compared to 62% of UCF(Fig. 12(c)). Thus, the ED metric is found to be superior than the existing metrics in terms of ranking the predictions for our proposed approach ALPM.

6 Discussions and Extensions

In this section, we discuss the limitations of our work and possible future directions of this work.

A location based social network (LBSN) does not provide a check-out facility. Therefore, while computing the time delay between any two consecutive check-ins

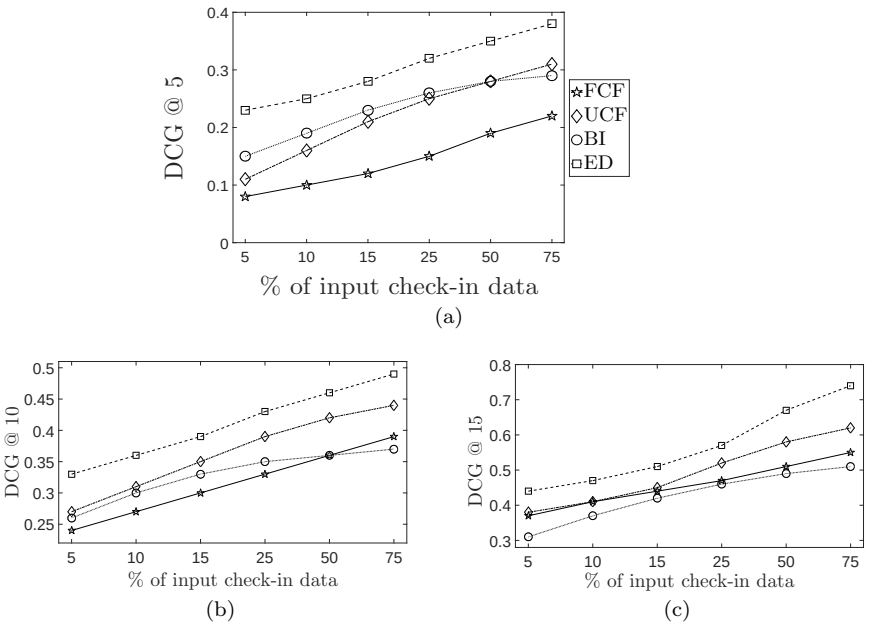

Fig. 12 *DCG* of our proposed ranking technique and the existing techniques for ALPM. (a) *DCG* of the top-5 predicted hidden locations vs the percentage of check-ins provided as input to ALPM. (b) *DCG* of the top-10 predicted hidden locations vs the percentage of check-ins provided as input to ALPM. (c) *DCG* of the top-15 predicted hidden locations vs the percentage of check-ins provided as input to ALPM.

$\langle l_i, l_{i+1} \rangle$ in Algorithm 1 (Section 4.1), we cannot separate the time spent by a user at l_i and the time taken by her to move from l_i to l_{i+1}. However, the time spent by an active user at a location is very important in context of selecting the potential consecutive check-in pairs having hidden locations. To learn the time spent by an active user at a location, we can proceed in the following direction.

Popular POIs and business houses maintain their own social network pages (Facebook, Twitter, Yelp, Foursquare) and official websites for their own benefit. These websites provide information such as detailed description of places, daily offers, reviews, ratings, opening and closing hours, show durations, average time spent by visitors, etc. This information can be utilized for estimating the check-out time. POIs such as museums, theatres, etc. have fixed show durations from which one can easily estimate the check-out time of a visitor. Other POIs such as restaurants, parks, etc. do not specifically have any fixed stay durations. However, it is observed that users often post review at POIs that include the time duration for which they stayed at a location. Therefore, exploring the user reviews using text mining algorithms can help to extract the time for which visitors generally stay at locations. In absence of such information at any particular location, we can classify the location into a category such as restaurant, theatre, museum, etc. and use the average check-out time of the category computed using the previous method.

Related to the user check-in data, there can be another interesting scenario. The user check-in data obtained from a LBSN can be obfuscated from a fine granular (raw) locations to coarse granular locations (region) due to privacy issues. In the current work, we considered location with fine granularity *i.e.* raw check-ins with specific latitude and longitude, and hence it cannot handle such a scenario. If granularity of a location is changed from fine to coarse, then multiple fine granular check-ins which are close to each other will be combined to form a single region. A fine granular check-in is selected as a representative point for such a region. These representative locations are considered as the coarse granular check-ins of an active user. To deal with such obfuscated user check-in data, one can incorporate the following two modifications in the proposed model.

1. **Conversion of fine check-ins to coarse check-ins:** The grouping of fine check-ins into coarse check-ins needs to be bounded by a distance and time threshold. We can adopt a strategy as described in [30] for such conversion. The coarse granular user check-in data is subsequently fed to the four modules of the proposed hidden location prediction model (ALPM).
2. **Entropy_Distance:** The entropy computation of a coarse granular location can be performed as the ratio of the number of times an active user has checked-in at the member locations of the region to the number of times all users have checked-in at the region.

In addition to the above scenarios, data privacy is another important issue which is closely related to our work. It has been observed that the LBSNs tend to apply various location privacy preserving techniques before publishing their data. Popular privacy preserving techniques like location obfuscation [35,36] and spatial cloaking [37,38] can be adopted by LBSNs to preserve location privacy. An important extension of this work would be to develop such a location prediction model which can perform well even if the available data is obfuscated or anonymized. The current approach by applying the above mentioned two changes can handle obfuscated check-in data. The spatial cloaking techniques use k-anonymity which ensures that at each location their exist atleast k number of users, so as to prevent re-identification of an individual. However, the proposed approach cannot handle public data which is anonymized using spatial cloaking techniques. Further research in this direction is believed to be a challenging task.

7 Conclusions

Users in LBSN deliberately hide visited locations. However, revealing unchecked location is an important part in many application domains. In this paper, we presented an approach which explores the association between location pairs, the unobserved locations using HMM and the proximal locations of each check-in for finding the hidden locations. Experimental results show that the proposed model outperforms the existing state of art technique in literature for an extracted LBSN check-in dataset of Beijing city, China. The proposed work can assist the GPS navigation system in cars for directing nearby POIs which are passed by, assist traffic modeling in populated areas, enhance various recommendation systems and others.

References

1. Justin Cranshaw, Eran Toch, Jason Hong, Aniket Kittur, and Norman Sadeh (2010) Bridging the gap between physical location and online social networks. In *Proceedings of the 12th ACM International Conference on Ubiquitous computing*, pp 119–128.
2. Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil (2011) An empirical study of geographic user activity patterns in foursquare. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, pp 70–573.
3. Mohamed Sarwat, Jie Bao, Ahmed Eldawy, Justin J Levandoski, Amr Magdy, and Mohamed F Mokbel (2012) Sindbad: a location-based social networking system. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp 649–652.
4. Lars Backstrom, Eric Sun, and Cameron Marlow (2010) Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web*, pp 61–70.
5. Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo (2011) Socio-spatial properties of online location-based social networks. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media*, pp 329–336.
6. Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra, and Karl Aberer (2013) Semantic trajectories: Mobility data computation and annotation. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):1–49.
7. Martin Erwig, Ralf Hartmut Gu, Markus Schneider, Michalis Vazirgiannis, et al. (1999) Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3(3):269–296.
8. Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo (2011) Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp 1046–1054.
9. Michael Fire, Lena Tenenboim-Chekina, Rami Puzis, Ofrit Lesser, Lior Rokach, and Yuval Elovici (2013) Computationally efficient link prediction in a variety of social networks. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):1–10.
10. Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma (2008) Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp 1–34.
11. Min Joong Lee and Chin Wan Chung (2011) A user similarity calculation based on the location for social network services. In *Proceedings of the 16th International Conference on Database Systems for Advanced Applications*, pp 38–52.
12. Eunjoon Cho, Seth A Myers, and Jure Leskovec (2011) Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp 1082–1090.
13. Adam Sadilek, Henry Kautz, and Jeffrey P Bigham (2012) Finding your friends and following them to where you are. In *Proceedings of the 5th ACM International Conference on Web search and Data mining*, pp 723–732.
14. Josh Jia Ching Ying, Wang Chien Lee, and Vincent S Tseng (2013) Mining geographic-temporal-semantic patterns in trajectories for location prediction. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):1–33.
15. Zheng Huo, Xiaofeng Meng, and Rui Zhang (2013) Feel free to check-in: Privacy alert against hidden location inference attacks in geosns. In *Proceedings of the 18th International Conference on Database Systems for Advanced Applications*, pp 377–391.
16. CC Robusto (1957) The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40.
17. Rakesh Agrawal, Tomasz Imieliński, and Arun Swami (1993) Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, 22(2):207–216.
18. Jiawei Han, Jian Pei, and Yiwen Yin (2000) Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, 29(2):1–12.
19. Sung Hyuk Cha (2007) Comprehensive survey on distance/similarity measures between probability density functions. *International Journal of Mathematical Models and Methods in Applied Sciences*, 1(4):300–307.
20. Lawrence R Rabiner (1989) A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, 77(2):257–286.

21. G David Forney Jr (1973) The viterbi algorithm. In *Proceedings of the IEEE*, 61(3):268–278.
22. Pierre A Devijver (1985) Baum's forward-backward algorithm revisited. *Pattern Recognition Letters*, 3(6):369–373.
23. Sherif Akoush and Ahmed Sameh (2007) Mobile user movement prediction using bayesian learning for neural networks. In *Proceedings of the 2007 International Conference on Wireless communications and Mobile computing*, pp 191–196.
24. Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang (2010) Collaborative location and activity recommendations with gps history data. In *Proceedings of the 19th International Conference on World wide web*, pp 1029–1038.
25. Jun Wang, Arjen P De Vries, and Marcel JT Reinders (2006) Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp 501–508.
26. Mao Ye, Peifeng Yin, and Wang-Chien Lee (2010) Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp 458–461.
27. Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic (2012) Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the 6th ACM Conference on Recommender systems*, pp 139–146.
28. Christopher J Burges, Robert Ragno, and Quoc V Le (2007) Learning to rank with nonsmooth cost functions. *Advances in Neural Information Processing Systems*, pp 193–200.
29. Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan (2009) Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp 621–630.
30. Pramit Mazumdar, Bidyut Kr Patra, Russell Lock, and Sathya Babu Korra (2016) An approach to compute user similarity for gps applications. *Knowledge-Based Systems*, 113:125–142.
31. Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási (2010) Limits of predictability in human mobility. *Science*, 327(5968):1018–1021.
32. Xin Lu, Linus Bengtsson, and Petter Holme (2012) Predictability of population displacement after the 2010 haiti earthquake. *Proceedings of the National Academy of Sciences*, 109(29):11576–11581.
33. Xin Lu, Erik Wetter, Nita Bharti, Andrew J Tatem, and Linus Bengtsson (2013) Approaching the limit of predictability in human mobility. *Scientific Reports*, 3(2923):1–9.
34. Michael J Shaw, Chandrasekar Subramaniam, Gek Woo Tan, and Michael E Welge (2001) Knowledge management and data mining for marketing. *Decision support systems*, 31(1):127–137.
35. Andreas Gutscher (2006) Coordinate transformation-a solution for the privacy problem of location based services? In *Proceedings of the 20th International Conference on Parallel and Distributed Processing*, pp 1–7.
36. Claudio Agostino Ardagna, Marco Cremonini, Ernesto Damiani, S De Capitani Di Vimercati, and Pierangela Samarati (2007) Location privacy protection through obfuscation-based techniques. In *Proceedings of the IFIP Annual Conference on Data and Applications Security and Privacy*, pp 47–60.
37. Marco Gruteser and Dirk Grunwald (2003) Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st International Conference on Mobile systems, applications and services*, pp 31–42.
38. Bugra Gedik and Ling Liu (2005) Location privacy in mobile systems: A personalized anonymization model. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, pp 620–629.