Exploiting Large-Scale Check-in Data to Recommend Time-Sensitive Routes

Hsun-Ping Hsieh[1], Cheng-Te Li[1], Shou-De Lin[1,2]
[1]Graduate Institute of Networking and Multimedia
[2]Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
{d98944006, d98944005, sdlin}@csie.ntu.edu.tw

ABSTRACT
Location-based services allow users to perform geo-spatial check-in actions, which facilitates the mining of the moving activities of human beings. This paper proposes to recommend time-sensitive trip routes, consisting of a sequence of locations with associated time stamps, based on knowledge extracted from large-scale check-in data. Given a query location with the starting time, our goal is to recommend a time-sensitive route. We argue a good route should consider (a) the popularity of places, (b) the visiting order of places, (c) the proper visiting time of each place, and (d) the proper transit time from one place to another. By devising a statistical model, we integrate these four factors into a goodness function which aims to measure the quality of a route. Equipped with the goodness measure, we propose a greedy method to construct the time-sensitive route for the query. Experiments on Gowalla datasets demonstrate the effectiveness of our model on detecting real routes and cloze test of routes, comparing with other baseline methods. We also develop a system *TripRouter* as a real-time demo platform.

Categories and Subject Descriptors
H.2.8 [**Database Management**]: Database Applications – *Data mining*.

General Terms
Algorithms, Management, Performance.

Keywords
Time-sensitive query, trip recommendation, check-in data.

1. INTRODUCTION
Location-based Services (LBS), such as Foursquare [1] and Gowalla[2], allow users to perform check-in actions that pin the geographical information of current location and time stamp onto their personal pages. The rapid accumulation of user check-in records can not only collectively represent the real-world human activities, but also serve as a great resource for location-based recommendation systems. Since the user-moving records implicitly reveal how people travel around an area with rich spatial and temporal information, including longitude, latitude, and check-in timestamp, one reasonable application leveraging such user-generated check-in data is to recommend the travel routes. Indeed, much existing work recommends routes using GPS trajectories [2][14] or geo-tagged photos [1][4][18].

In this paper, instead of purely relying on past moving trajectories to recommend traveling paths, we propose a novel time-sensitive trip route recommendation framework that takes advantage of the user-check-in data. We argue that a proper route recommendation system should consider the following factors when designing a route:

- **The popularity of a place.** Popular landmarks by definition should attract more visitors.
- **The proper time to visit a place.** In general, the pleasure of visiting a place can be significantly diminished if arriving at the wrong time. Some places have a wider range of visiting time while others are constrained to certain particular time slots. For example, most people do not want to visit a beach during boiling hot noon, but rather arrive in the late afternoon to enjoy the sunset scene. Or certain ball game events usually take place at particular time period (e.g. in the evening). As shown in Figure 1, as derived from the Gowalla check-in data described in Section 5, visitors visit some places with higher probability during certain time slots. For example, people usually visit the Empire State Building from about 12:00 to the mid night (note that this place is famous for its excellent night view), (b) people tend to visit the Madison Square Garden in the early evening for a basketball game, (c) the proper time to visit the Central Park is during daytime, and (d) Time Square is preferred from afternoon to midnight.
- **The amount of time required to transit from one place to another.** The transit time between places is highly correlated to visiting the next places at proper time. To find the next place with the proper visiting time, we should consider the amount of time spent on traveling from the current location to the next. For example, one has bought tickets to a football game at a stadium 2 hours away. He will logically choose to start traveling toward the stadium 2 hours ahead of the official kick off time instead of going to a nearby museum 30 minutes away.
- **The visiting order of places.** The visiting order of places is important as it depends on the nature of places and human preference. For example, going to the gym first then going to nearby restaurant for dinner is a better plan than the other way around since it is unhealthy to exercise right after a meal.

While some places are extremely sensitive to the visiting time, the others (e.g. movie theaters) might not possess such strict constraint. An intelligent route recommendation system should

UrbComp'12, August 12, 2012. Beijing, China.

[1] Fouresquare: https://foursquare.com/

[2] Gowalla: http://gowalla.com/

consider such diversity and be able to create a route that has higher chance of satisfying users' needs. This paper argues that by exploring the check-in data, it is possible to design a statistical model to achieve such goal.

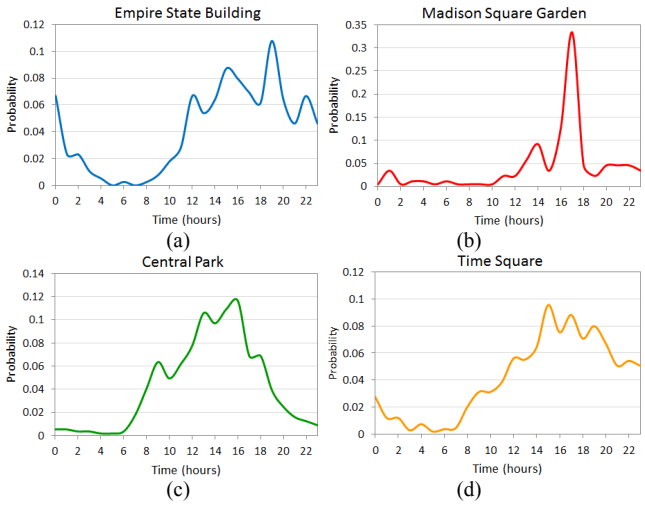

Figure 1: The distribution of the visiting probability at each time unit (hour) for (a) Empire State Building, (b) Madison Square Garden, (c) Central Park, and (d) Time Square. These distributions are derived from the Gowalla check-in data.

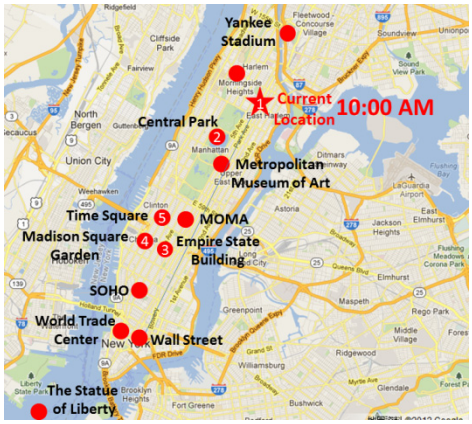

Figure 2: An illustration of recommending a trip route for a query located at the star position with the time stamp 10:00 AM, in Manhattan area, New York City. The goal is to find a trip route connecting some check-in locations with the consideration of the mentioned four requirements.

The online check-in data provide plenty of explicit or implicit information that allows us to fulfill the abovementioned requirements for the sake of planning a proper trip route. First, we can derive from the check-in data the number of people who have visited a certain place, and thus derive the popularity of places. Second, users in LBS tend to perform check-in actions to keep track of their trips in traveling days. As a result, we can obtain and consider the visiting order of places. Third, the check-in records contain the visiting time stamps of locations. Users in LBS are able to collectively reveal the proper visiting time of places. Fourth, followed by the check-in time stamps from existing routes, we are able to have the transit time between places. Equipped with such elements, we utilize the check-in data to recommend the trip routes. Let us use Figure 2 as an example to elaborate the major idea of our time-sensitive trip route recommendation. Assuming a certain user starts to travel from his New York City hotel, marked with a star in Figure 2, at 10:00 AM. There are several popular attractions he/she can visit in a day, including the four famous places mentioned in Figure 1. If the user wants to visit all four places, a possible trip route consists of going to Central Park first, followed by Empire State Building, Madison Square Garden, and finally Time Square.

Formally, the goal of this paper is to construct a time-sensitive route from the check-in data. Given a starting location with a time stamp as the time-sensitive location query, we propose to find a sequence of check-in locations as the trip route, in which each location can be visited at the proper time with the proper transit time from one place to another in the route. The benefit of such time-sensitive trip route is three-fold. First, the user can maximize the capability/price value on visiting each place. Second, with the suggested transit time, users are able to control their schedule more accurately and manage their time effectively. Third, the trip planner can recommend users one or more attractions along the way.

We propose a statistical approach with a greedy search algorithm to construct the time-sensitive routes with respect to the query. The method consists of two phases. In phase one, we measure the quality of a route by devising a goodness function, which integrates the abovementioned four requirements. In phase two, with the query location and time, we greedily find next visiting places by optimizing the goodness function.

We summarize the contributions of this paper in the following.

- We propose a novel time-sensitive trip route recommendation problem using the check-in data in location-based services. We fulfill the idea by developing a *TripRouter* system based on the real-world Gowalla check-in data.
- Conceptually, we argue that a good route should consider four elements: (a) the popularity of a place, (b) the visiting order of places, (c) the proper visiting time of a place, and (d) the proper transit time between places.
- Technically, we devise a goodness function to measure the quality of a route. By exploiting some statistical methods, we model the four requirements of a good route into the design of the goodness function. In addition, for the given time-sensitive location query, we develop a greedy algorithm to search for the route by optimizing the goodness function.

This paper is organized as follows. We describe the related work in Section 2. Section 3 devises the goodness of a route and elaborates the greedy route search algorithm. We evaluate the proposed method in Section 4 and demonstrate the *TripRouter* system in Section 5. Section 6 concludes this work.

2. RELATED WORK

Route Planning by GPS Trajectory Data. There is lots of related work about route planning using the GPS trajectories. J. Juan et al. [14] [15] find the fastest routes to a destination. Z. Chen et al. [2] and L.-Y. Wei et al. [11] search for popular and attractive trajectories for recommendation. Z. Chen et al. [3] find the top-k trajectories connecting some user-given locations. H. Yoon et al. [13] and Y. Zheng et al. [17] propose the itinerary recommendation by considering user preference based on mined trajectory attributes. Y. Zheng et al. [16] [17] aim to discover interesting and classical travel sequences. L.-A. Tang et al. [10] finds the top-k nearest neighboring trajectories with the minimum aggregated distance to some query locations. L.-Y. Wei et al. [12]

construct the top-k routes which sequentially pass through the query locations within the specified time span. Though there are many successful proposals to solve different kinds of route planning problems, the issues of proper visiting time of places and proper transit time between places are never tackled. To achieve such goal, this work proposes to generate the time-sensitive trip routes using check-in data.

We use Table 1 to summarize the differences between our work and other relevant studies. Here we list some important issues about route planning, including: whether it allows the Query of certain Locations (QL), and whether it considers the following ideas: Popularity (PO), Visiting Order (VO), Visiting Time (VT), Transit Time (TT), User Preference (UP), Distance (DI), Travel Duration (TD), and Top-*k* retrieval (TK).

Table 1: Summarization of differences between this paper and other related work.

	QL	PO	VO	VT	TT	UP	DI	TD	TK
[14][15]		■	■			■	■	■	
[2]	■	■	■						
[11]		■	■						■
[3]	■		■						■
[13]	■	■	■		■	■		■	■
[16][17]	■	■	■			■			■
[10]	■						■		■
[12]	■	■						■	■
This work	■	■	■	■	■			■	

Route Recommendation Using Social Media. The rapid rise of social media applications generates huge-volume geo-spatial data of human activities, such as geo-tagged photos in Flickr and check-in records in Foursquare. Both geo-tagged photos and check-in data can reveal how people sequentially visit places in an area. Using geo-tagged photos, Y. Arase et al. [1] mine frequent route patterns for recommendation. A.-J. Cheng et al. [4] propose personalized travel recommendation based on personal profiles and visual attributes of geo-tagged photos. X. Lu et al. [7] and T. Kurashima et al. [6] construct routes based on user preference of must-go destinations, visiting time, and travel duration. Z. Yin et al. [18] mine and rank trajectory patterns from geo-tagged photos and diversify the ranking results. L.-Y. Wei et al. [12] infer the top-*k* routes traveling a given location sequence within a specified travel time from uncertain check-in data. Different from these work, we aim to perform knowledge discovery to construct the time-sensitive routes.

3. METHODOLOGY

3.1 Basic Definitions

Definition 1: Location. A location l_i is a tuple, $l_i = (x_i, y_i)$, where x_i is the longitude, y_i is the latitude.

Definition 2: Route of Check-in Locations. A route is a sequence of locations with the corresponding time stamps, denoted by s, $s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>$, where n is the number of locations. Throughout this paper, we focus on recommending single-day route, which implies t_n-t_1 is no more than 24 hours.

Definition 3: Time-sensitive Query. We define the *Time-sensitive Query* as $Q = (l_q, t_q)$, where l_q is the initial location of a user, and t_q is the starting time for this trip.

Definition 4: Time-sensitive Route. Given a time-sensitive query, we define the output *Time-sensitive Route* as a sequence of check-in locations $s_r = <(l_1,t_1), (l_2,t_2), ..., (l_k,t_k)>$, where $l_1 = l_q$, $t_1 = t_q$, and k is the number of locations in the route, which can be either specified by users or determined using existing time constraint of the trip.

In the following we will describe how to measure the quality of a time-sensitive trip route. Based on the proposed goodness definition, we are able to search and recommend better time-sensitive routes given an initial time-sensitive query.

3.2 Measuring the Quality of a Trip Route

In order to construct a high-quality route for recommendation, we need to first design a proper metric to measure the quality of any given route. We propose that a good trip route should consider the following four factors: (a) the popularity of a place, (b) the proper visiting time of a location, (c) the proper transit time traveling from one location to another, and (d) the visiting order of places in the route. We attempt to model these factors into the goodness function, and utilize such function to greedily selecting locations for the construction of the final trip route.

3.2.1 Route Popularity

A popular place, by definition, should be somewhere that attracts more visitors in general. If a route contains more popular places, it has higher potential to satisfy a user. The popularity of a place can be represented by the number of check-in actions performed at that place. In our goodness measure of a route, we first consider the popularity of places in the route. We define the relative popularity of a location l_i as:

$$pop(l_i) = \frac{N(l_i)}{N_{max}}$$

where $N(l_i)$ is the number of check-in of the location l_i, and N_{max} is the maximum number of check-in among all the locations in the check-in data. Given a route $s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>$, we define the popularity-based goodness function $f_{pop}(s)$ as:

$$f_{pop}(s) = \left(\prod_{i=1}^{n} pop(l_i)\right)^{\frac{1}{n}}$$

3.2.2 Proper Visiting Time

The check-in data reveals that while some locations (e.g. park and movie theater) are popular regardless of the visiting time in a given day, other locations (e.g. stadium and beach) are more attractive during certain time period of the day. We propose to learn such time-dependent popularity of each location from the check-in data. We begin from defining the *Temporal Visiting Distribution* as the following.

Definition 5: Temporal Visiting Distribution (TVD) of a Location. We define a *Temporal Visiting Distribution* for a location l, $TVD_l(t_i)$, as the probability distribution of a randomly picked check-in record of l occurs at time t_i. For example, in a 24-hour span, *TVD* can be a legal probability distribution shown in Figure 3. *TVD* can easily be learned from check-in data, representing how popular a place is at a given time.

Using *TVD*, we can determine whether it is proper to visit a place at a given time. For example, assuming we want to know how well a decision is to visit a place at 8:00AM, given the location's *TVD* is represented as the green dotted line in Figure 3. To do that, we propose to first generate a thin Gaussian distribution $G(t;\mu,\sigma^2)$ whose mean value μ is 8 with a very small variance σ^2 (e.g. standard deviation is 1). And then we can transform the original task into measuring the difference between the Gaussian distribution with the learnt *TVD* of such location. Here we use the

symmetric Kullback-Leibler (KL) Divergence between $G(t;\mu,\sigma^2)$ and $TVD_l(t)$ to represent the fitness of the assignment. The formal mathematical definition of a fitness score between a place l and a time t can be defined as

$$D_{KL}(G(t;\mu,\sigma^2)||TVD_l(t))$$
$$= \sum_x G(x;\mu,\sigma^2)\, log\, \frac{G(x;\mu,\sigma^2)}{TVD_l(x)} + \sum_x TVD_l(x)\, log\, \frac{TVD_l(x)}{G(x;\mu,\sigma^2)}$$

Conceivably, a smaller KL value indicates better match between the assignment and the distribution learned from data.

Consequently, we formally define the temporal visiting goodness function $f_{visit}(s)$ of a route s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, as a combination of the popularity of places together with the fitness of each location over time, in the following equation.

$$f_{visit}(s) = \left(\prod_{i=1}^{n} D_{KL}\left(G(t;t_i,\sigma^2)||TVD_{l_i}(t)\right) \times \frac{1}{pop(l_i)}\right)^{\frac{-1}{n}}$$

If the places in a route s are visited during the proper time period, the $f_{visit}(s)$ value would become higher.

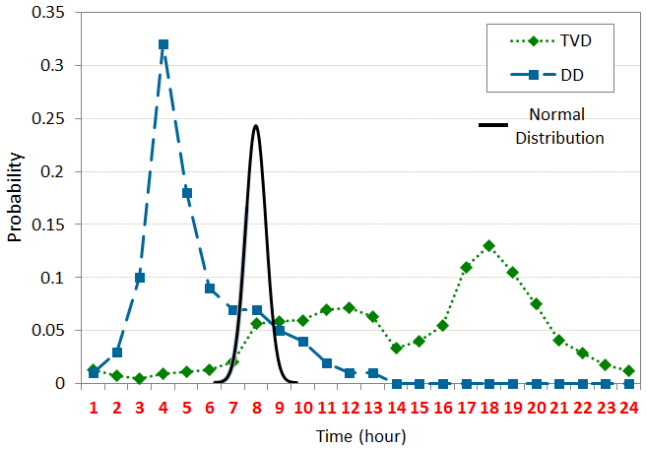

Figure 3: Examples of the temporal visiting distribution (TVD) (the green dotted curve) for a certain location l_i, and the duration distribution (DD) (the blue dashed curve) between location l_i and l_j. The black solid curve represents a normal distribution of a particular time assignment to measure the fitness values.

3.2.3 Proper Transit Time Duration

To schedule a good trip route, another key element to be considered is the visiting time of each place as well as the transit time from one place to another. Although the check-in data cannot explicitly tell us the above two kinds of information, we can simply treat the duration between two checked-in places as the summation of the visiting time of the first place plus the transportation time from the first to the second place. Such duration can further be utilized to evaluate the quality of a trip. Here we propose the *Duration Distribution*, as defined in the following, to model such 'visiting plus transit time' between places.

Definition 6: Duration Distribution (DD) between Two Locations. We define the *Duration Distribution (DD)* between locations l_i and l_j as the probability distribution over time duration t, $DD_{l_i l_j}(t)$, which can be obtained from the following random experiment: randomly pick two consecutive check-in records (l_i, t_i), (l_j,t_j) of a person, and calculate the probability that t_j-t_i=t.

Again, we consider only one-day trip, and therefore treat the outcome space of *DD* between hours 0 through 24. For example, any legal probability distribution between hours 0 through 24 can be a *DD* (e.g. the blue dashed line in Figure 3).

Similar to what we do to *TVD*, given a pair of locations l_i and l_j together with an assignment of a given duration Δ among them, we can model Δ as a thin Gaussian distribution and compare it with $DD_{l_i l_j}(\Delta)$ using symmetric KL divergence. Consequently, for a route s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, it is possible to know how good the route is based on the durations between places by defining a goodness function of duration:

$$f_{duration}(s) = \left(\prod_{i=1}^{n-1} D_{KL}\left(g(t;\Delta_{i,i+1};\sigma^2)||LTD(l_i,l_j)\right)\right)^{\frac{-1}{n-1}}$$

A route s with higher value of $f_{duration}(s)$ indicates such route can be visited with proper "transit+staying" time between places.

Here we use Figure 4 as an illustration to summarize our idea of utilizing *TVD* and *DD* to measure the goodness of a trip route. Given a route s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>. We use symmetric KL divergence to measure the visiting fitness of each location l_i by calculating a $D_{KL}(l_i)$ value between TVD_{li} and a narrow Gaussian distribution. We also use KL divergence to measure the fitness of each transition $l_i \rightarrow l_j$ and derive a $D_{KL}(\Delta_{ij})$ between $DD_{l_i l_j}$ and a thin Gaussian distribution. Eventually we compute the geometric mean of such D_{KL} values to be the time-related route goodness.

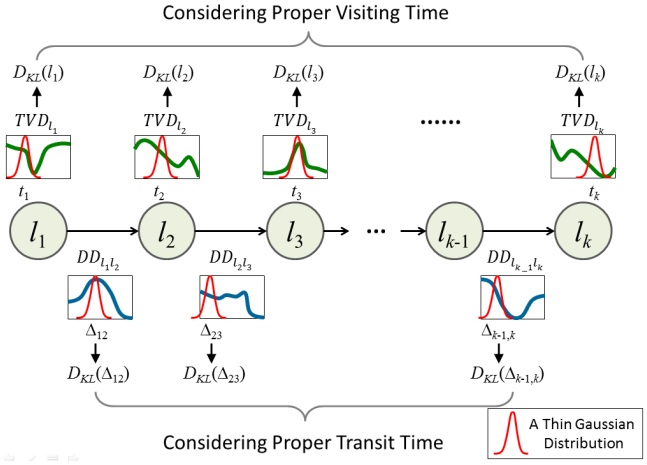

Figure 4: For a route s=<(l_1,t_1), (l_2,t_2), ..., (l_k,t_k)>, we compute 2*k*-1 values of KL-divergence and then take the geometric mean of such values as the time-dependent goodness of a route.

3.2.4 Proper Visiting Order

Due to the characteristic of each place, there might be certain latent patterns about the order of the places to be visited. With the check-in data, we are able to learn such orders and exploit them to evaluate the quality of a route. For example, going to restaurant for dinner and then going back to hotel is better than the other way around. In this section, we propose to exploit the idea of the *n*-gram language model to measure the quality of the order of visits in a trip route. Using the check-in corpus, we can first generate the n-gram probabilities of locations. Then, given a route s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>, we can compute its *n*-gram

probability. We consider such n-gram probability as the goodness of visiting order. Technically, we use the average value of the probabilities of uni-gram, bi-gram, and tri-gram to estimate the goodness of orders. Note that the uni-gram probability is corresponding to the popularity-based route goodness. We can formally write the probabilities as follows.

$$P_{uni}(s) = f_{pop}(s)$$

$$P_{bi}(s) = (P(l_1)P(l_2|l_1)P(l_3|l_2)\cdots P(l_n|l_{n-1}))^{\frac{1}{n}}$$

$$P_{tri}(s) = (P(l_1)P(l_2|l_1)P(l_3|l_1l_2)\cdots P(l_n|l_{n-2}l_{n-1}))^{\frac{1}{n}}$$

Therefore, the goodness of visiting order of a route can be defined:

$$f_{order}(s) = \frac{\left(P_{uni}(s) + P_{bi}(s) + P_{tri}(s)\right)}{3}$$

Higher $f_{order}(s)$ value represents better quality of route. Note that we utilize the add-one technique for smoothing.

3.2.5 *Final Goodness Function*

Here we integrate the goodness measures of the proper visiting time, the proper transit time duration, and the proper visiting order into the final goodness function $f(s)$. The final goodness function contains two parts. The first part is the average value of the temporal visiting goodness $f_{visit}(s)$ and the location transition goodness $f_{duration}(s)$. The second part is the visiting order goodness $f_{order}(s)$. We use a parameter $\alpha \in [0,1]$ to devise a linear combination of such two parts. The final goodness function $f(s)$ is defined in the following.

$$f(s) = \alpha \times \left(\frac{f_{visit}(s) + f_{duration}(s)}{2}\right) + (1-\alpha) \times f_{order}(s)$$

A route s with higher value of $f(s)$ will be considered as a better route. Experiments in Section 5.3 suggest $\alpha \approx 0.9$ being more effective on measuring route quality. Such result exhibits the usefulness of the proposed time-sensitive route recommendation.

Algorithm 1. *TimeRoute* algorithm

Input: (a) $RouteDB$: routes extracted from the check-in data;
(b) $Q = (l_q, t_q)$: the time-sensitive location query;
(c) k: the number of locations in the final route.
Output: a time-sensitive route $s_r = \langle (l_1, t_1), (l_2, t_2), \ldots (l_k, t_k) \rangle$.

```
1:  s_r = ⟨(l_1 = l_q, t_1 = t_q)⟩.
2:  for i = 2 to k do:
3:    C_i = {l_c | l_{i-1} → l_c in RouteDB}.
4:    f_max = 0.
3:    for each l_c ∈ C_i do:
4:      s_tmp = s_r + ⟨(l_c, t_c)⟩.
5:      Compute the goodness f(s_tmp).
6:      if f(s_tmp) > f_max do:
7:        s_r = s_tmp.
8:        f_max = f(s_tmp).
9:  Return: s_r.
```

3.3 Greedy Algorithm *TimeRoute*

In this section, we formally describe the problem of time-sensitive trip route recommendation based on the proposed goodness measure. And then we propose a greedy algorithm, *TimeRoute*, to construct the time-sensitive routes for a given query.

Problem Definition. Given (a) the routes extracted in the check-in data, (b) the time-sensitive location query $Q = (l_q, t_q)$, and (c) the number k of locations in the final route, the goal is to construct a route $s_r = \langle (l_1=l_q, t_1=t_q), (l_2,t_2), \ldots, (l_k,t_k) \rangle$ to optimize $f(s_r)$.

To solve this problem, we devise a greedy algorithm, *TimeRoute*, to achieve the local-optimal solution. The basic idea is to select next place based on the goodness function $f(s)$. Starting from the query location (line 1 in Algorithm 1), when selecting next location l_i ($i > 2$), we first identify a set of candidate locations C_i by collecting locations which have been ever followed by l_i (line 3). Then for each location in the candidate set C_i, we select the candidate l_c with the maximum goodness value given the existing route, and append it to the final route s_r. (line 4-8). Such procedure will terminate when k spots are identified in the route.

4. EXPERIMENTS

4.1 Dataset and Data Analysis

We utilize the Gowalla dataset [5] which has been exploited for location-based analysis in several places, such as [8] and [9]. The Gowalla dataset contains 6,442,890 check-in records from Feb. 2009 to Oct. 2010. The total number of check-in locations is 1,280,969. Considering a route as a sequence of check-in locations of a user within a day, we construct the route database *RouteDB* containing 2,605,867 routes, among them 1,469,130 has only length one and are not used. The average route length is 4.09, without considering length-1 routes. Figure 5 shows the distribution of the route length, which is highly-skew and heavily-tailed. Figure 5 also shows that people usually do not prefer visiting too many locations in a day, but with some exceptions. Figure 6 shows the distribution of the time duration between two places. It indicates that people consider places closer to where they are when they are planning the next destination.

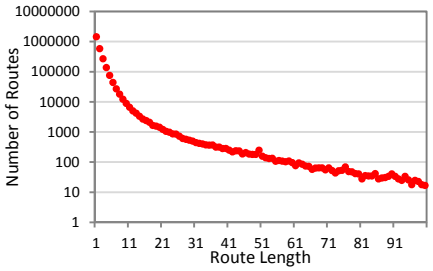

Figure 5: Distribution of route length in *RouteDB*.

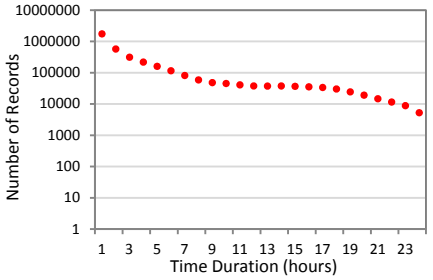

Figure 6: Distributions of time duration in *RouteDB*.

Table 2: The statistics of *RouteDB* and the three subsets.

	Total Number of Check-ins	Avg. Route Length	Variance of Route Length	Distinct Check-in Locations
RouteDB	6,442,890	4.09	48.04	1,280,969
New York	103,174	4.46	71.24	14,941
San Francisco	187,568	4.09	58.36	15,406
Paris	14,224	4.45	75.73	3,472

From *RouteDB*, we extract three subsets of the check-in data, which corresponds to cities of New York, San Francisco, and Paris. Some statistics are reported in Table 2. We can find the average route lengths and their variance in New York and Paris are significantly longer than average. Figure 7 shows the distribution of route length in the three subsets of check-in data while Figure 8 shows the distribution of the time duration.

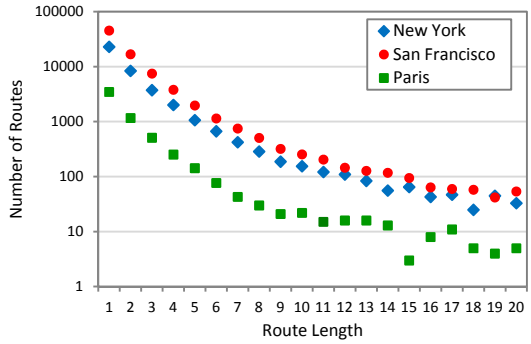

Figure 7: Distribution of route length for three cities (hour).

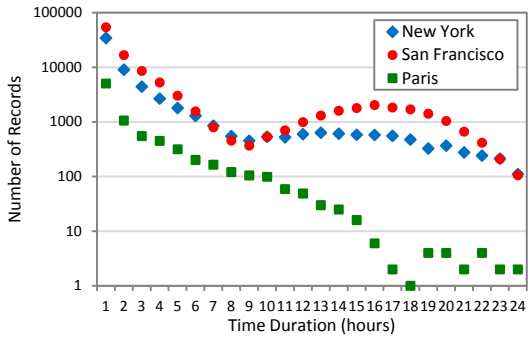

Figure 8: Distribution of time duration for three cities (hour).

4.2 Evaluation Plan

In this section, we introduce two evaluation plans for verifying the performance of our proposed method using the data in these three cites, and compare the results with several baseline methods.

Experiment 1: **Pair-wise Time-sensitive Route Detection.** In this experiment, we would like to verify whether our goodness model can rank the existing routes higher than the non-existing ones. We first randomly choose one thousand real routes from the check-in data. Note that the time stamp is associated with each location *l*. For each route, we replace a portion of the locations with other locations in the city to generate a pseudo route. To make the task non-trivial, we adopt a replacing strategy to replace a location with a 'plausible' one instead of a randomly selected one. That is, to replace a location at position *i* of a route, we only choose from candidate locations that once appear right after the location at position *i*-1 (e.g. the bigram probability of them is non-zero) instead of simply picking a random location. Furthermore, after the replacement, we want to make sure the generated pseudo routes do not exist in the database. That is, there is no such route in the database of the same location sequences together with the same associated time stamps. As can be seen in Figure 11 to 13, the amount of replaced locations varies from 10% to 50% of the total number in a route. We then use our fitness model to examine each pair of the existing route and its pseudo route, and record how frequently our method ranks the correct one higher. Finally, we report the accuracy of our method and compare it with the baseline results. The accuracy is calculated as the number of successfully detected routes divided by the number of pair instances.

Similarly, we can generate another kind of pseudo route by perturbing the time stamps of certain locations in an existing route. For example, given an existing route s=<(l_1,t_1), (l_2,t_2),… (l_{i-1}, t_{i-1}), (l_i,t_i), (l_{i+1},t_{i+1}),..., (l_n,t_n)>, we change t_i to a different time t_j, where $t_{i-1} < t_j < t_{i+1}$. We expect a proper fitness function to assign lower score to such pseudo routes.

Experiment 2: Time-sensitive Cloze Test of Locations in Routes. Given some real trip routes with time stamp in each location, by removing some middle locations, the goal is to test whether a method can successfully identify the removed location. Higher hit rate indicates better quality of recommendation.

Baseline Approaches. To evaluate the effectiveness of our method, we design the following four baseline methods for both experiment 1 and experiment 2.

- **Distance-based Approach.** This method chooses the closest location to the current spot as the next spot to move to. It rates a route using the goodness function $f_d(s) = (\prod_{i=1}^{n} \frac{1}{D(l_i,l_{i-1})})^{\frac{1}{n}}$, where $D(l_i, l_{i-1})$ is the geographical distance between two consecutive locations.
- **Popular-based Approach.** This method chooses the most popular spot of a given time in that city as the next spot to move to. It rates the path using the goodness function $f_{pop}(s)$ as have been defined previously in Section 3.2.1.
- **Forward Heuristic Approach.** The forward heuristic chooses a location l_i that possesses the largest bi-gram probability with the previous location $P(l_i|l_{i-1})$ as the next location to move to. Its goodness function is $f_{forw}(s) = P_{bi}(s)$, as defined previously in section 3.2.4.
- **Backward Heuristic Approach.** The backward heuristic chooses a location l_i that possesses the largest bi-gram probability with the next location $P(l_i|l_{i+1})$ as the next location to move to. The fitness function can be described as $f_{backw}(s) = (P(l_1|l_2)P(l_2|l_3)\cdots P(l_{n-1}|l_n))^{\frac{1}{n}}$.

4.3 Experimental Results

Section 4.3.1 shows the results of Experiment 1 and Section 4.3.2 illustrates the outcome of Experiment 2. For both experiments, we implement four baseline methods to compare with our proposed *TimeRoute* method.

4.3.1 Pairwise Time-Sensitive Route Detection

In experiment 1, we first vary the number of replaced locations from 10% to 50% and report the accuracy of different methods. Figure 9 contains the results for New York City. Our fitness model can achieve around 97% accuracy in distinguishing the real routes from replaced ones. The accuracy scores of the forward and backward heuristics vary from 89% to 93%. The popular-based and distance-based methods do not do a good job here. Similar trend happens in Paris (Figure 11), but for San Francisco (Figure 10), our method shows much higher accuracy comparing with others. The results are not surprising because our method does consider the location preference over time and location order.

Figure 12 shows the results of creating pseudo paths by shifting time stamp for some locations. Again we vary the ratio of change from 10% to 50%. The results show that our model can almost perfectly detect such change, better than the popularity-based

method (around 15% accuracy). The other competitors do not have the capability to distinguish such pairs because they do not consider time information during route generation, and therefore the fitness scores are identical for such pair of routes.

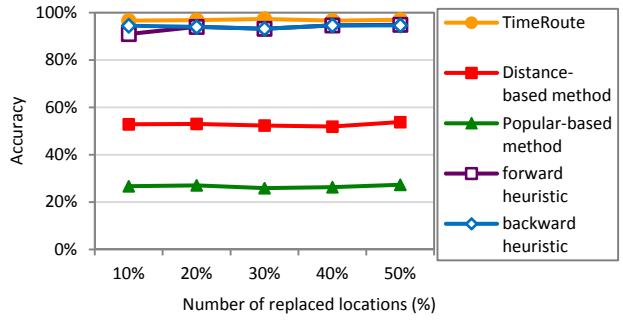

Figure 9: Accuracy by varying the number of replaced locations in New York.

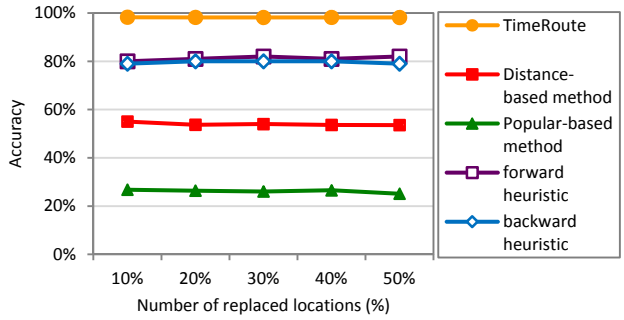

Figure 10: Accuracy by varying the number of replaced locations in San Francisco.

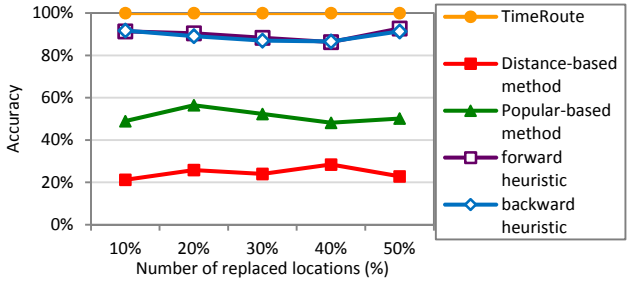

Figure 11: Accuracy by varying the number of replaced locations in Paris.

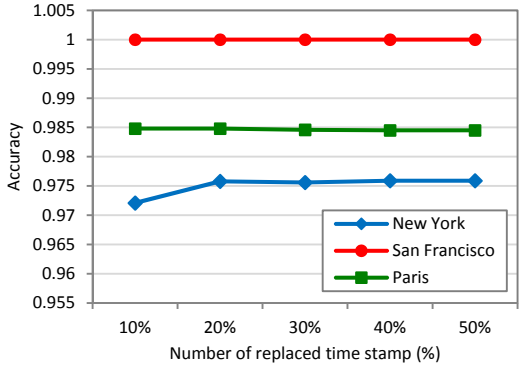

Figure 12: Accuracy by varying the number of replaced time stamp for our method in the three cities.

4.3.2 Time-Sensitive Cloze Test in Routes

In cloze experiment of locations in routes, we calculate hit rate by varying the position of missing location. Generally speaking, the preceding position of missing location obtains lower hit rate than latter one because the system can generally do better when more information is revealed.

As reported in Figure 13-15, the hit rates of the four baseline models are often lower than 10% in these three cities, while we can achieve 15%~40% hit rate.

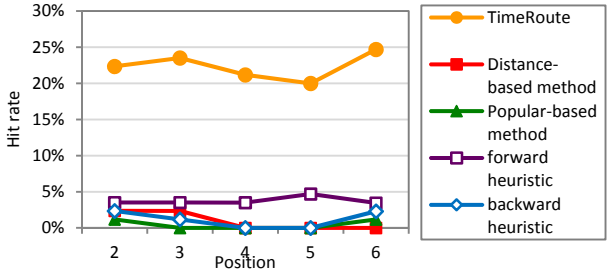

Figure 13: Accuracy by varying the position of missing location in New York.

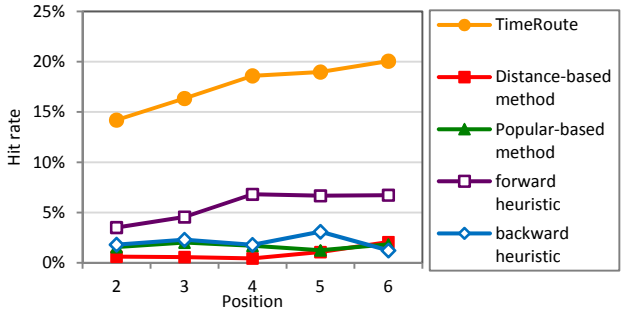

Figure 14: Accuracy by varying the position of missing location in San Francisco.

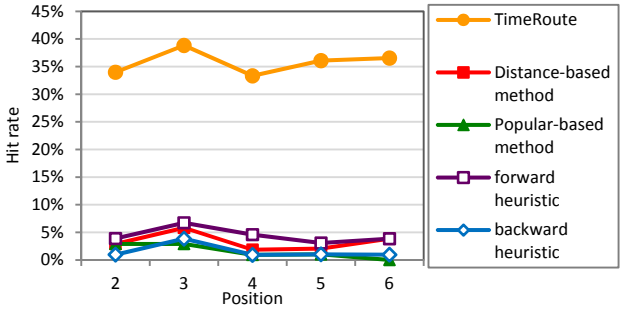

Figure 15: Accuracy by varying the position of missing location in Paris.

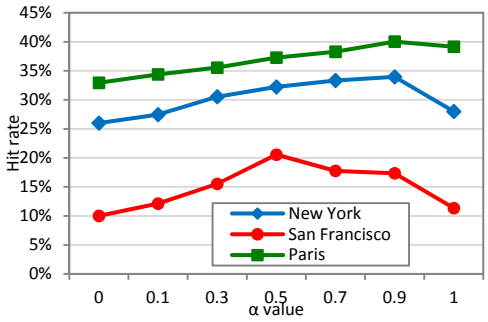

Figure 16: The impact of α on the time-sensitive cloze test for the three cities.

Impact of α: Next, we examine how sensitive our model is to the parameter α, ranging from 0 to 1. We use the hit rate of cloze test and the results are shown in Figure 16. In New York and Paris, the best α value is around 0.9. That is, much more weight is assigned to time-sensitive models than the visiting order on cloze test task. In San Francisco City, α performs well while varying from 0.5 to 0.9.

5. SYSTEM DEMONSTRATION

Using our model, we develop an online time-sensitive trip route recommendation system, called *TripRouter*. The system snapshot is shown in Figure 17. Users first determine the city they want to travel, and then select one location as their starting location, together with the starting time. *TripRouter* also allows users to specify their estimated travel time duration and the desired number of locations of such trip. We list the three major functions of *TripRouter* as below: (a) time-sensitive route recommendation, (b) displaying diverse information of locations and routes such as location attributes, route statistics, and some geo-tagged photos obtained from Flickr, and (c) recommending the transportation mode by querying Google Map API according to mined transit time duration.

Below we show three recommended routes querying from Central Park at different starting time, where the route length *k* is set as 4.

- **Central Park at 9PM:** Central Park (9AM) → New York City Center (11AM) → 5th Ave (5PM) → FAO Schwarz restaurant (7PM).
- **Central Park at 2PM:** Central Park (2PM) → The Museum of Modern Art (3PM) → Bergdorf Goodman (4PM) → Lee's art shop (7PM).
- **Central Park at 5PM:** Central Park (5PM) → 5th Ave (6PM) → Pulitzer Fountain (7PM) → Four season hotel (8PM).

The above examples tell us that our *TripRouter* system is able to recommend the best route based on the specified time and location.

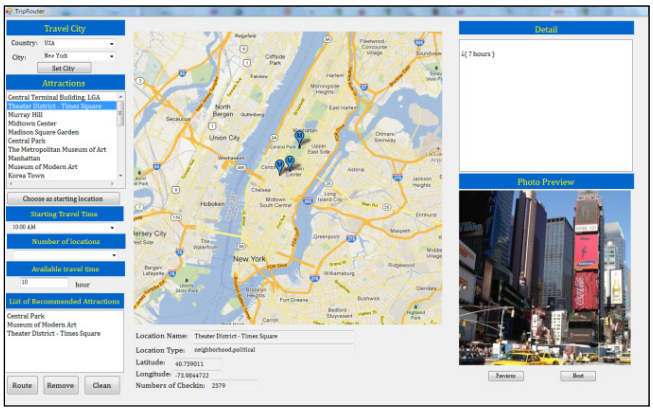

Figure 17: The system interface of *TripRouter*.

6. CONCLUSION

This paper tries to address an important research question: how much the check-in data can provide in terms of designing a suitable trip route. The solution provided by us seems to be very encouraging as it shows that one can indeed squeeze a lot of knowledge from check-in data to design a time-sensitive trip route that has higher potential of satisfying the users. Note that our approach is mostly data-driven, which assures diverse results can be learned from different cities in which visiting patterns may vary with different culture and characteristics of the city. Ongoing work focuses on two directions: using maximum likelihood estimator to accurately model the visiting time duration of a place and transportation time between places, and further exploit the collaborative filtering approaches to take advantage of the user and location similarities.

7. REFERENCES

[1] Y. Arase, X, Xie, T. Hara, and S. Nishio. Mining People's Trips from Large Scale Geo-tagged Photos. In *ACM MM* 2010.

[2] Z. Chen, H. T. Shen, and X. Zhou. Discovering Popular Routes from Trajectories. In *IEEE ICDE* 2011.

[3] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching Trajectories by Locations: An Efficiency Study. In *ACM SIGMOD* 2010.

[4] A.-J. Cheng, Y.-Y. Chen, Y.-T. Huang, W. H. Hsu, and H.-Y. M. Liao. Personalized Travel Recommendation by Mining People Attributes from Community-Contributed Photos. In *ACM MM* 2011.

[5] E. Cho, S. A. Myers, and J. Leskovec. Friendship and Mobility: User Movement in Location-based Social Networks. In *ACM KDD* 2011.

[6] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel Route Recommendation Using Geotags in Photo Sharing Sites. In *ACM CIKM* 2010.

[7] X. Lu, C. Wang, J.-M. Yang, Y. Pang, and L. Zang. Photo2trip: Generating Travel Routes from Geo-tagged Photos for Trip Planning. In *ACM MM* 2010.

[8] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo. Socio-spatial Properties of Online Location-based Social Networks. In *ICWSM* 2010.

[9] S. Scellato, A. Noulas, C. Mascolo. Exploiting Place Features in Link Prediction on Location-based Social Networks. In *ACM KDD* 2011.

[10] L.-A. Tang, Y. Zheng, X. Xie, J. Yuan, X. Yu, and Jiawei Han. Retrieving k-Nearest Neighboring Trajectories by a Set of Point Locations. In *SSTD* 2011.

[11] L.-Y. Wei, W.-C. Peng, B.-C. Chen, and W.-C. Peng. PATS: A Framework of Pattern-Aware Trajectory Search. In *MDM* 2010.

[12] L.-Y. Wei, Y. Zheng, and W.-C. Peng, Constructing Popular Routes from Uncertain Trajectories. In *ACM KDD* 2012.

[13] H. Yoon, Y. Zheng, X. Xie., and W. Woo. Social Itinerary Recommendation from User-generated Digital Trails, In *Personal and Ubiquitous Computing*, 2011

[14] J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with Knowledge from the Physical World. In *ACM KDD* 2011.

[15] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-Drive: Driving Directions Based on Taxi Trajectories. In *ACM SIGSPATIAL GIS* 2010.

[16] Y. Zheng., L. Zhang, X. Xie, and W.-Y. Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *WWW* 2009.

[17] Y. Zheng and X. Xie. Learning Travel Recommendations from User-generated GPS Traces. In *ACM TIST* 2011.

[18] Z. Yin, L. Gao, J. Han, J. Luo, and T. Huang. Diversified Trajectory Pattern Ranking in Geo-tagged Social Media. In *SDM* 2011.