TripRouter: A Time-Sensitive Route Recommender System

Hsun-Ping Hsieh, Cheng-Te Li, Shou-De Lin
Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan
{d98944006, d98944005, sdlin}@csie.ntu.edu.tw

***Abstract*—Location-based services allow users to perform geo-spatial recording actions, which facilitates the mining of the moving activities of human beings. This paper proposes a system, *TripRouter*, to recommend time-sensitive trip routes consisting of a sequence of locations with associated time stamps based on knowledge extracted from large-scale location check-in data. We first propose a statistical route goodness measure considering: (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. Then we construct the time-sensitive route recommender with two major functions: (1) constructing the route based on the user-specified source location with the starting time, (2) composing the route between the specified source location and the destination location given a starting time. We devise a search method, *Guidance Search*, to derive the routes efficiently and effectively. Experiments on Gowalla check-in datasets with user study show the promising performance of our *TripRouter* system.**

Keywords—trip route; time-sensitive; check-in data; location-based services.;

I. INTRODUCTION

Location-based Services (LBS), such as Foursquare and Gowalla, allow users to perform the action of location recording that pins the geographical information of current locations and time stamps onto their personal pages. By continuously recording such actions by users, a location sequences dataset can be generated. The rapid accumulation of location sequence data can not only collectively represent the real-world human activities, but also serve as a handy resource for constructing location-based recommendation systems. Since the user-moving records implicitly reveal how people travel around in an area with rich spatial and temporal information, including longitude, latitude, and recording timestamp, one reasonable application leveraging such user-generated location sequence data is to construct and recommend travel routes. Indeed, many of existing works had recommended routes using GPS trajectories (e.g. [1][2][10]). Furthermore, using 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. [3] mine frequent route patterns for recommendation. A.-J. Cheng et al. [4] propose personalized travel recommendation using geo-tagged photos. X. Lu et al. [5] construct routes based on user preference querying locations. Zheng et al. [6] present the *activity trajectory similarity search* which returns *k* check-in trajectories that cover the activity labels. Lu et al. [7] develop a personalized trip recommendation that scores attractions by social links and temporal properties. L.-Y. Wei et al. [1] infer the top-*k* detailed routes traveling a given location sequence within a specified travel time. Different from these works, we aim to perform knowledge discovery to construct the time-sensitive routes.

We use Table I to summarize the differences between our work and other relevant trip recommendation system. 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 I. SUMMARIZATION OF DIFFERENCES BETWEEN THIS PAPER AND OTHER RELATED WORKS.

	QL	PO	VO	VT	TT	UP	DI	TD	TK
[10]		■	■			■	■	■	
[2]		■			■	■		■	■
[3]	■						■		■
[4]	■	■	■						
[5]	■	■	■					■	
[6]	■					■			■
[7]	■	■							■
[1]	■	■						■	■
This work	■	■	■	■	■			■	

In this paper, instead of relying on past trajectories to recommend trip routes, we propose a novel time-sensitive route recommender system, *TripRouter*, using location check-in data. We argue that a good route should consider four factors. (a) The popularity of a place: popular landmarks will likely attract more visitors. (b) The proper time to visit a place: the pleasure of visiting a place can be significantly diminished if arriving at the wrong time. Some places have a wider range of preferred 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. Sports game events usually take place at particular time period. (c) The amount of time transiting from one place to another: for example, if one has bought tickets to a football game at a stadium 2 hours away, then he or she shall 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 then. (d) The visiting order of places: for example, going to the gym first then going to restaurant for dinner might be a better plan than the other way around since it is not healthy to exercise right after a meal.

We use check-in data to acquire the time-stamped geographical information in *TripRouter*. Check-in data provides explicit or implicit information that allows us to fulfill the abovementioned requirements for the sake of planning a proper trip route. First, we can distill 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. 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 hypothesize the transit time between places. Equipped with such elements, we utilize the check-in data to recommend trip routes and construct our *TripRouter* system.

Formally, the goal of this work is to recommend time-sensitive routes using time-stamped location sequence data according to user requirements. We propose to tackle two real-world demands of recommending time-sensitive routes, which corresponds to the two main functions in our *TripRouter* system. The first is to construct a time-sensitive route given a *source* location, and the second is to create a time-sensitive route given the *source-destination* pair of locations. Both queries consider the starting time of the trip. Given a source or source-destination query, our system will return a sequence of recommended places as the final route, in which each location can be visited at a proper time with a reasonable transit time from one place to another in the route. In addition, in the query, we also allow users to determine the extent of time-sensitivity of locations through specifying a time-sensitivity parameter. Time-sensitive routes are supposed to be more effective than a simple route without time stamp as it allows the users to better manage their time during the trip. Both queries are very common for real-world trip planning.

In *TripRouter*, we propose a statistical approach to model the time-sensitivity of location, and a novel search algorithm to recommend time-sensitive routes with respect to the queries. In general, our work consists of two important issues. First, we aim to design a goodness function, which integrates the abovementioned four requirements about a good trip route to measure the quality of a route. Second, given a query, we devise an effective and efficient search method, *Guidance Search*, to identify the places to be visited by optimizing the route goodness function.

II. *TripRouter* SYSTEM

Notations. A location l_i is a tuple, $l_i = (x_i,y_i)$, where x_i is the longitude, y_i is the latitude. A route is a sequence of locations with the corresponding time stamps, denoted by $s=<(l_1,t_1),(l_2,t_2),...,(l_n,t_n)>$, where n is the number of locations. The *source query* $Q_s=(l_s,t_s)$ contains a starting location l_s with time stamp t_s, and the *source-destination query* $Q_d=(l_s,t_s,l_d)$ further contains a destination location l_d, where k is the number of locations in the final route (either specified by users or determined automatically). A *time-sensitive route* is $s_r = <(l_1,t_1),...,(l_k,t_k)>$, where $l_1 = l_q$, $t_1 = t_q$, and/or $l_k=l_d$.

***Time-sensitive Route Construction* problem.** Given (a) routes derived from location check-in data, and (b) either the source query $Q_s=(l_s,t_s)$ or the source-destination query $Q_d=(l_s,t_s,l_d)$, the goal is to construct a route $s_r=<(l_1=l_s,t_1=t_s),...,(l_k,t_k)>$ to optimize the time-sensitive route goodness function $f(s_r)$. A route with maximum route goodness score tends to be a preferred one. Note that l_k is required to be l_d for the source-destination query.

A. Measuring Route Goodness

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.

1) Route Popularity: 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 recording actions performed at that place. Given a route $s=<(l_1,t_1),(l_2,t_2),...,(l_n,t_n)>$, we define the popularity-based goodness function $f_{pop}(s) = (\prod_{i=1}^{n} pop(l_i))^{-n}$, where $pop(l_i) = N(l_i)/N_{max}$, $N(l_i)$ is the number of recording actions performed on location l_i, and N_{max} is the total number of recording actions among all locations.

2) Proper Visiting Time: We define a Temporal Visiting Distribution for a location l, TVD_l (t_i), as the probability distribution of a randomly picked recording action of location l occurs at time t_i. Then we can determine whether it is proper to visit a place at a given time. We generate a thin Gaussian distribution $G(t;\mu,\sigma^2)$ whose mean value μ is 8 with a small variance σ^2 (e.g. standard deviation is 1), and measure the difference between the Gaussian distribution with the learnt TVD of such location by symmetric Kullback-Leibler (KL) Divergence. Consequently, the temporal visiting goodness function $f_{visit}(s)$ of a route $s=<(l_1,t_1), (l_2,t_2), ..., (l_n,t_n)>$ is defined as a combination of the popularity of places together with the fitness of each location over time:

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

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

3) Proper Transit Time: We treat the duration between two checked-in places as the summation of the visiting time of the first place plus the transportation time from one place to another. To model such 'visiting plus transit time' between places, we propose the Duration Distribution (DD) between locations l_i and l_j, which is defined as the probability distribution over time duration t, $DD_{l_i l_j}(\Delta)$, and can be obtained from the following random experiment: randomly pick two consecutive location recording actions (l_i,t_i), (l_j,t_j) of a person, and calculate the probability that t_j-t_i=t. We consider only one-day trip, and therefore treat the outcome space of DD between hours 0 through 24. Given a pair of locations l_i and l_j together with an assignment of a given duration Δ among them, we model Δ as a thin Gaussian distribution and compare it with $DD_{l_i l_j}(\Delta)$ by symmetric KL divergence. Given a route s, the goodness function of durations is defined by:

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

4) Visiting Order: We exploit n-gram language model to measure the goodness of the order of visits in a trip route from the location sequence corpus. Technically, we use the average value of the probabilities of uni-gram, bi-gram, and tri-gram to estimate the goodness of orders: $f_{order}(s) = \left(P_{uni}(s) + P_{bi}(s) + P_{tri}(s)\right)/3$. Higher $f_{order}(s)$ value represents better quality of route. Note that we utilize the add-one technique for smoothing.

B. Final Goodness Function

We divide the final goodness function into two parts and provide a weighting parameter $\alpha \in [0,1]$ for users to determine the significance and balance of such two parts. The first part is the average temporal visiting goodness $f_{visit}(s)$ and the location transition goodness $f_{duration}(s)$. The second part is visiting order goodness $f_{order}(s)$. The final goodness function $f(s)$ is defined as

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

A route s with higher value of $f(s)$ is considered as a better route. Note that the parameter α provides users the flexibility to specify whether they prefer the time-sensitive routes.

C. Route Construction Algorithm

We develop the *Guidance Search* algorithm to recommend time-sensitive routes for the source and source-destination queries. *Guidance Search*, consisting of two parts, is a kind of *best-first search* that finds a least-cost path from a given initial node to the goal. The first is the *heuristic satisfaction function*, which is in charge of the guidance to determine the next most promising location towards the destination. The second is the *backward checking mechanism* that keeps the search tree (i.e., all the expanded routes starting from the source location) for reconstructing the route with higher *satisfaction* score.

We consider the route goodness function $f(s)$ to design the heuristic satisfaction function $f^*(l)$, which is to measure the satisfaction of selecting location l as the next visiting location considering the sub-route from source location l_s to location l (i.e., $l_s \rightarrow l$) and from location l to destination location l_d (i.e., $l \rightarrow l_d$). Therefore, we design $f^*(l)$ to have two parts: (1) time-sensitive route goodness function $g(l) = f(l_s \rightarrow l)$, and (2) *heuristic function* $h(l)$ that considers both the time-sensitive goodness $f(l \rightarrow l_d)$ and the geographical distance $d(l \rightarrow l_d)$ of a sub-route from location l to destination l_d. We use the geographical distance between l and l_d as the steering force to direct the search process to move towards the destination location. When selecting next visiting location during the route construction, those locations with shorter distance to the destination has higher chance to be picked, if the rest of the criteria are equally satisfied. Moreover, because there could be many sub-routes from l to l_d in the route database, we will compute all the scores and take the best one as the final $h(l)$ value. Consequently, the heuristic function is formally written as:

$$h(l) = max_{(l \rightarrow l_d) \in S(l \rightarrow l_d)}\{\sqrt{f(l \rightarrow l_d) \times (1 - d(l \rightarrow l_d))}\}$$

where $S(l \rightarrow l_d)$ is the set of sub-routes starting from l to l_d. Eventually, the final heuristic satisfaction function is defined as:

$$f^*(l) = (1-\beta) \times g(l) + \beta \times h(l),$$

where $\beta \in [0,1]$ is the parameter to control the strength of the guidance to the destination. Higher β indicates stronger guidance. When $\beta = 0$, $f^*(l)$ is simply a greedy search with the backward checking mechanism, and can be used to tackle the source query (because in $g(l)$ function, no destination information is needed). Note that β is the most important parameter in our system, which allows users to determine the time-sensitivity of locations in the final routes, according to user needs and scenarios.

The *backward checking mechanism* is the key to the best-first search in our algorithm. When exploiting the *heuristic satisfaction function* to choose the next location to visit, it is necessary to expand all neighbor locations to generate the satisfaction scores. We keep track of such scores in the search tree. When it is needed to select the next visiting location, not only the expanded locations from the current location, but also those had ever been expanded during the previous rounds can be considered. In other words, in addition to continue expanding the current location, the algorithm can possibly go backward to consider the previously expanded nodes for finding the ones with the highest satisfaction score.

We elaborate the details of *Guidance Search* algorithm as follows. We first construct the initial route s_0 by including the source location l_s. A priority queue is employed for the purpose of the backward checking mechanism. Each element in the priority queue consists of a route s and the corresponding heuristic satisfaction score. The priority queue automatically sorts its elements according to their satisfaction scores. We add s_0 to initialize the priority queue. After setting the final route s_r as the initial one s_0, we perform the iterative expansion search process until the route s_r is constructed up to length k. For each iteration, the last location l_{last} in the route s_r with the highest satisfaction score is identified and each possible next visiting location l_{next} is put into a candidate set C. Then for each candidate of the next location l_c, we can derive the heuristic satisfaction score $f^*(l_c)$ (if $l_d = \emptyset$, we set the weighting parameter $\beta = 0$ for the function f^*; otherwise $0 < \beta \leq 1$). We put $f^*(l_c)$ together with the corresponding route s_{tmp} into the priority queue. The priority queue will then pick the next best route and location to conduct the further expansions. Finally, the route s_r is reported as the final route.

III. EVALUATION

We use Gowalla dataset [9] to construct our *TripRouter* system. Such dataset contains 6,442,890 check-in records from Feb. 2009 to Oct. 2010 on 1,280,969 locations. By regarding a route as a sequence of check-in locations of a user within a day, we have the route database containing 1,136,737 routes whose lengths are more than one (the average length of them is 4.09). We extract three subsets for the experiments, which corresponds to cities of New York (NY), San Francisco (SF), and Paris (PR). We test the effectiveness of our goodness model and demonstrate the performance of the proposed search methods by a *time-sensitive location cloze test*. Given some real trip routes with time stamp in each location, by removing some middle locations, the goal of cloze experiment is to test whether a method can successfully identify the removed locations. We use *Hit Rate* as the accuracy measure. Given *N* removals of locations over all routes, and assumed *M* places out of *N* is successfully predicted, the hit rate is defined as *M*/*N*. Higher hit rate indicates better quality. We compare our method with a strong greedy algorithm [8] and a series of baseline methods: (1) Distance-based approach: choose the closest location to the current spot as the next one. (2) Popular-based approach: choose the most popular spot of a given time

in that city as the next one. It rates the path using the goodness function $f_{pop}(s)$. (3) Forward heuristic: choose the location possessing the largest bi-gram probability with the previous location $P(l_i|l_{i-1})$ as the next location. (4) Backward heuristic: choose the location possessing the largest bi-gram probability with the next location $P(l_i|l_{i+1})$ as the next one.

In the cloze experiment, we vary the number of guessing instance per route and report the hit rate in three cities. Here we set $\beta = 0.5$ (i.e., $g(l)$ and $h(l)$ are considered as equally important). The results are shown in Fig. 1. In general, the hit rate of each method is decreasing while the number of guessing instance increases. Nevertheless, our method significantly outperforms the greedy search method [8] and other baselines.

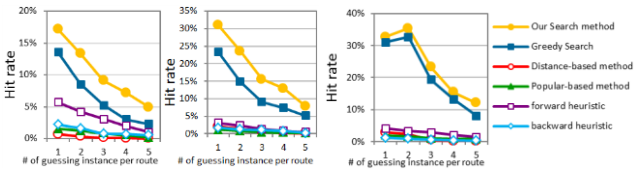

Fig. 1. Hit rates by varying the number of guessing instance per route (i.e., 1, 2, 3, 4, and 5) on SF, NY, PR (left to right).

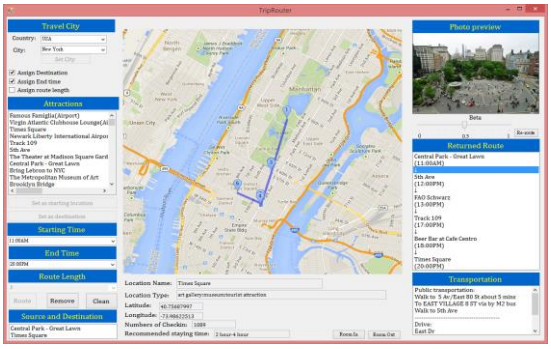

Fig. 2. The system snapshot of *TripRouter*.

IV. System Functions and Demonstration

The snapshot of *TripRouter* is shown in Fig. 2. The main function of *TripRouter* allow users to specify the city they intent to travel, select one location with starting time as the source, and/or the destination location, and/or the desired number of places during a trip. In, addition, users can determine the extent of location time-sensitivity by specifying the β parameter. We list four additional functions: (a) show the pictures of locations obtained from Flickr, (b) recommend the proper stay time on locations, (c) provide the transportation according to the mined transit time duration, and (d) display contextual location information, including popularity, category, nearby events and restaurants, and weather.

Below we demonstrate an example by varying β as 0, 0.5 and 1 to show three recommended routes querying from Central Park to Time Square starting at 10AM, in which the route length *k*=4. The results are highlighted in blue and shown in Fig.3.

Route 1. ($\beta = 0$): *Central Park (10AM) → Metropolitan Museum of Art (1PM) → American Museum of Natural History (4PM) → Time Square (9PM).*

Route 2. ($\beta = 0.5$): *Central Park (10AM) → 5th Ave (1PM) → New York Public Library (5PM) → Time Square (7PM).*

Route 3. ($\beta = 1$)*: Central Park (10AM) → New York Public Library (2PM) → Bryant Park (7PM) → Time Square (9PM).*

Fig. 3. Recommended routes for $\beta = 0$, 0.5, and 1 from left to right.

For $\beta = 0$, since it pays attention to fit the time-sensitive score, $g(l)$, it produce longer transit time to the final destination. Thus, we think that $\beta = 0$ provides routes for users who have much traveling time, and really care about time-sensitivity of locations. On the contrary, $\beta = 1$ finds shorter route to the destination, and thus it is suitable for travelers who prefer to visit more places with a short period. Moreover, $\beta = 0.5$ is suitable for general travelers because it tries to strike a balance between $g(l)$ and $h(l)$.

V. Conclusion

We develop the *TripRouter* system that measure and recommend time-sensitive trip routes based on user needs on source, destination, current time, and time-sensitivity. Experimental results and user study encourage the promising performance and practicability of *TripRouter*. *TripRouter* 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. Moreover, despite that we emphasized on the check-in data, in fact any kinds of route data (e.g. GPS trajectory data) can be exploited in our framework.

References

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

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

[3] Y. Arase, X, Xie, T. Hara, and S. Nishio. Mining People's Trips from Large Scale Geo-tagged Photos. *ACM MM* 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. *ACM MM* 2011.

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

[6] K. Zheng, S. Shang, J. Yuan, and Y. Yang. Towards Efficient Search for Activity Trajectories. *IEEE ICDE* 2013.

[7] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng. Personalized Trip Recommendation with Multiple Constraints by Mining User Check-in Behaviors. *ACM GIS* 2012.

[8] H.-P. Hsieh, C.-T. Li and S.-D. Lin, Recommending Time-Sensitive Routes by Exploiting Large-Scale Check-in Data. *ACM UrbComp* 2012.

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

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