KSTR: Keyword-aware Skyline Travel Route Recommendation

Yu-Ting Wen*, Kae-Jer Cho*, Wen-Chih Peng*, Jinyoung Yeo†, Seung-won Hwang‡
*National Chiao Tung University, Taiwan
†Pohang University of Science and Technology, Republic of Korea
‡Yonsei University, Republic of Korea
{ytwen,kjcho13,wcpeng}@cs.nctu.edu.tw,jinyeo@postech.ac.kr,seungwonh@yonsei.ac.kr

***Abstract*—With the popularity of social media (e.g., Facebook and Flicker), users could easily share their check-in records and photos during their trips. In view of the huge amount of check-in data and photos in social media, we intend to discover travel experiences to facilitate trip planning. Prior works have been elaborated on mining and ranking existing travel routes from check-in data. We observe that when planning a trip, users may have some keywords about preference on his/her trips. Moreover, a diverse set of travel routes is needed. To provide a diverse set of travel routes, we claim that more features of Places of Interests (POIs) should be extracted. Therefore, in this paper, we propose a Keyword-aware Skyline Travel Route (*KSTR*) framework that use knowledge extraction from historical mobility records and the user's social interactions. Explicitly, we model the "Where, When, Who" issues by featurizing the geographical mobility pattern, temporal influence and social influence. Then we propose a keyword extraction module to classify the POI-related tags automatically into different types, for effective matching with query keywords. We further design a route reconstruction algorithm to construct route candidates that fulfill the query inputs. To provide diverse query results, we explore Skyline concepts to rank routes. To evaluate the effectiveness and efficiency of the proposed algorithms, we have conducted extensive experiments on real location-based social network datasets, and the experimental results show that *KSTR* does indeed demonstrate good performance compared to state-of-the-art works.**

I. Introduction

Location-based social network services allow users to perform check-in and share their check-in data with their friends. In particular, when a user is traveling, the check-in data are in fact a travel trajectory with some photos and tag information. As a result, a massive number of trajectories are generated, which play an essential role in many well-established research areas, such as mobility prediction, urban planning and traffic management. In this paper, we focus on trip planning and intend to discover travel experiences from shared data in location-based social networks. To facilitate trip planning, the prior works in [1][2][3][4][5] provide an interface in which a user could submit the query region and the total travel time. In contrast, we consider a scenario where users specify their preferences with keywords. For example, when planning a trip in Sydney, one would have "Opera House". As such, we extend the input of trip planning by exploring possible keywords issued by users. However, the query results of existing travel route recommendation services usually rank the trajectories simply by the popularity or the number of uploads of trajectories.

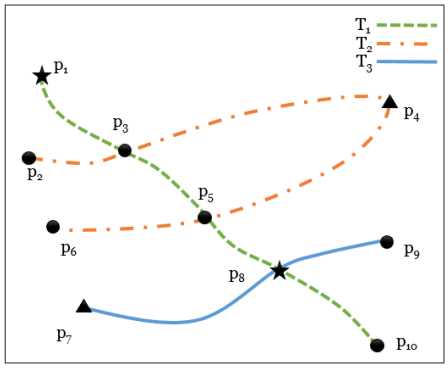


Fig. 1. Keyword-aware travel routes query running example.

TABLE I. Example of trajectory dataset

Tid	Uid	Pid	keyword	time	POI score vector
T_1	u_1	p_1	Opera House	10:00	(0.04, 0.2)
T_1	u_1	p_3	Bar	12:00	(0.25, 0.2)
T_1	u_1	p_5	Bar	15:30	(0.2, 0.8)
T_1	u_1	p_8	Opera House	17:30	(0.04, 0.3)
T_1	u_1	p_{10}	Bar	19:00	(0.04, 0.2)
T_2	u_2	p_2	Bar	10:30	(0.02, 0.2)
T_2	u_2	p_3	Bar	12:30	(0.25, 0.2)
T_2	u_2	p_4	Sunset	17:00	(0.05, 0.2)
T_2	u_2	p_5	Bar	19:00	(0.2, 0.8)
T_2	u_2	p_6	Bar	19:30	(0.25, 0.8)
T_3	u_3	p_7	Sunset	18:30	(0.4, 0.8)
T_3	u_3	p_8	Opera House	19:30	(0.04, 0.3)
T_3	u_3	p_9	Bar	20:00	(0.1, 0.1)

For such ranking, existing works [6][7][8] derive a scoring function, and each trajectory will have one score according to its features (e.g., the number of Places of Interest, the popularity of places). Usually, the query results will have similar trajectories. In contrast, this work considers the diversity of results, as high scoring trajectories are often too similar to each other. Our goal is to retrieve a greater diversity of trajectories based on the travel factors considered.

In this paper, we develop a Keyword-aware Skyline Travel Route (*KSTR*) framework to retrieve several recommended trajectories where keyword means the personalized requirements users have for the trip. Consider an example illustrated in Figure 1, the related route information of which is stored in Table I. For ease of illustration, each POI is associated with one keyword (though our model can support multiple keywords) and a two-dimensional score vector (each dimension repersents the rank of a feature). Assume a tourist plans a date with a set

of keywords ["Whisky" "Sydney Cove" "Sunset"]. First, we can find that these keywords vary in their semantic meaning: "Sydney Cove" is a geographical region; "Sunset" is related to a specific time period (evening) and locations such as beach; "Whisky" is the attribute of POI.

We argue knowing semantics is important, as some query keywords do not need to be matched in POI keyword. For example, p_9, even though its name does not include "Whiskey", is a good match, as it is an important attribute of Bar POIs. Similarly, "Sydney Cove" is not mentioned, but based on the location of Opera House, p_8 matches the requirement. As a result, T_3 matches all the requirements, which could not be supported by existing simple keyword-based matches.

With a set of travel routes, feature scoring should be considered to find proper recommendations. We explore three travel factors: "Where: people tend to visit popular POIs", "When: each POI has its proper visiting time", and "Who: people might follow social-connected friends' footsteps". In the view of POI, we store the attractiveness score and the visiting time information in the POI score vector. In the view of user, we also consider a score to quantify individual's influence in recommendation.

Additionally, we have mentioned that the final results may have similar characteristics and be monotonous due to that all factoring are aggregated into one score for each travel route. Consequently, the system will retrieve top-1 or top-K trajectories with the highest score as the results. Users may not understand the characteristic of these trajectories through the final one score (e.g., Which one has most interesting landmarks? Which one is well-connected to the place I want to go?) so that it may be hard to choose a trajectory from the final results. Furthermore, user need to pre-defined the weight for each factor although it is hard to select suitable weight in most cases. Since travel route recommendation has to take several factors into consideration to emphasize the unique travel factors of travel routes, we borrow the concept of Skyline to retrieve travel routes. Skyline search on the travel routes retrieves all the possible optimal results in terms of features derived, and hence a diverse set of travel routes is retrieved. Consider an example in Figure 1, where the score vector of POIs represents the attractiveness score and the visiting time information. To compute the average POI score of T_1, T_2 and T_3, we get the final score values (0.1,0.34), (0.15,0.44), and (0.18,0.3) respectively. The skyline result is $\{T_2, T_3\}$.

Consequently, in this paper, travel routes have several score values mined from social media. By exploiting Skyline query, a diverse set of travel routes is determined. In addition, travel routes could be constructed from different trajectory segments. We further propose one approximate algorithm to efficiently derive travel routes.

The contributions of this paper are summarized as follows:

- We propose a *KSTR* framework in which users are able to issue a set of keywords and a query region, and for which query results contain diverse trip trajectories.
- We propose a novel keyword extraction module to extract three types of keywords: Geo-specific keywords, Temporal keywords and Attribute keywords.
- We extract three travel factors of POIs from LBSNs: attractiveness of POIs, the visiting time of POIs and the geographical social influence.
- We propose a trajectory reconstruction method to partition trajectories into segments by considering spatial and temporal features.
- Skyline query for travel route search is adopted to combine the multi-dimensional measurements (POI attractiveness, proper visiting time and geographical social influence) of routes, which increase the diversity of the recommended results.
- An approximate algorithm is adopted to derive efficient results for the online interactive system.

To evaluate our proposed framework, we conducted experiments on real LBSN and photo datasets. The experimental results show that *KSTR* is able to retrieve travel routes that are of interest to users.

II. Framework Overview

In this section, the proposed framework *KSTR* is presented. As can be seen in Figure 2, *KSTR* is comprised of two modules: the offline pattern discovery and scoring module and the online travel routes exploration module.

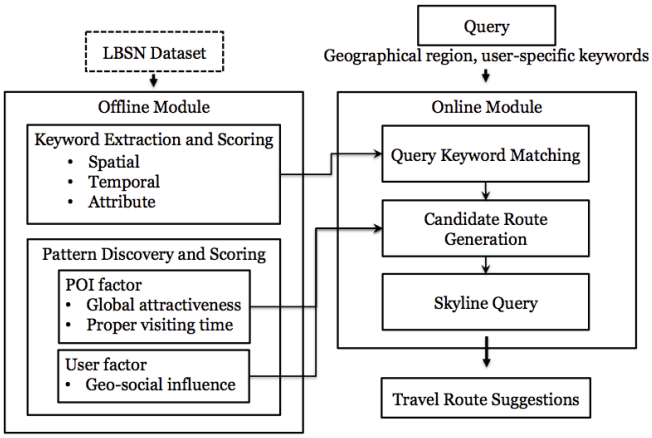


Fig. 2. The overview of the proposed *Keyword-aware Skyline Travel Route* framework

Offline pattern discovery and scoring module: Given an LBSN dataset, we first analyze the tags of each POI to determine the semantic meaning of the keywords, which are classified into (i) Geo-specific keywords, (ii) Temporal keywords and (iii) Attribute keywords according to their features. Then, we define the attractiveness scores and the proper visiting time of the POIs. For this goal, we translate the trajectories into a region transition graph, in which the vertices represent POIs and edges indicate the transition relationships among POIs. Furthermore, we determine the proper visiting time of each POI by analyzing the population distribution with timestamps, and assigning a time score for each time interval. Besides the global features that produce the same results for different people, social influence considers the geo-social networks for each user. In short, the main task of the offline module is to decide a set of POIs and derive their scores for each feature.

Online travel routes exploration module: In this module, we provide an interface for users to specify query ranges and preference-related keywords. With the map interface provided, users could simply input their query range as a rectangle, decide the time period to stay, and add keywords for personalized preference. Once it receives a specified range and time, the online module will retrieve those trajectories that overlap the query range and reconstruct proper travel routes in the stay time period. Then, the online module will compute a matched score of how well the trajectory is connected to the keywords. Consequently, the online module returns the most representative trajectories considering the aforementioned three scores to the users in our trajectory search Web interface.

III. Keyword Extraction and Pattern Discovery

This section describes an offline process of the scoring mechanism for keywords and POIs and pattern discovery from trajectory histories.

A. Keyword Extraction

In this subsection, we present how we extract the semantic meaning of the keywords and propose a matched score to describe the degree of connection between keywords and trajectories. The keyword extraction module first computes the spatial, temporal and attribute scores for every keyword w in the corpus. At query time, each query keyword will be matched to the pre-computed score of matching w.

1) Geo-specific keywords: Some tags are specific to a location, which represents its spatial nature. To quantify the geo-specificity of a tag, an external database identifies geo-terms in the overall tag set and then the tag distribution on the map rates the identified geo-terms. Specifically, to identify name tags, we leverage an external geo-database. In Microsoft Bing services, Geocode Dataflow API (GDA) [1] can query large numbers of geo-terms and their representative locations. For a tag w, using GDA, we set $GDA(w)$ as 1 if its location is returned, and 0 otherwise. Then, using the geographic distribution of the tags, we can find place-level geo-terms like 'Taipei101' in noisy geo-terms. Country-level geo-terms like 'USA' and city-level geo-terms like 'Seattle' are far more widely distributed on the globe than place-level geo-terms. Thus, we compute the variance $GeoVar(w)$ of the $(latitude, longitude)$ set including a tag w. With these features, we define a geo-specificity (GS) score of a tag w as:

$$GS(w) \propto GDA(w) \cdot exp(-GeoVar(w)) \tag{1}$$

We consider a tag w as a geo-specific keyword if $GS(w)$ is greater than a pre-defined threshold.

2) Temporal keywords: Some tags are specific to a time interval, which represents its temporal nature. To quantify the temporal-spatiality of a tag, time distribution on a tag rates the identified temporal-terms. Using time distribution of tags, we can find tags associated with a specific time interval like 'sunset'. Tags independent of time like 'Taipei' are far more widely distributed in time than time-specific tags. Thus, to identify temporal-tags, we compute the variance $TimeVar(w)$ of the creation time of check-ins including a tag w. With these features, we define a temporal-specificity (TS) score of a tag w as:

$$TS(w) \propto exp(-TimeVar(w)) \tag{2}$$

We consider a tag w as a temporal keyword if $TS(w)$ is greater than a pre-defined threshold. Then, given a temporal keyword w, we generate a 2-dimensional Gaussian $\mathcal{N}_t(\mu, \sigma^2)$ that models the distribution of the occurring time of w and define the associated time of w as a time interval with up to two standard devations from μ.

3) Attribute keywords: To find attribute keywords, we consider tags frequently associated with a POI (TF), while not with so many other POIs (IDF). To quantify the relevance between a tag and a POI, we define a "document" as an estimated check-in set I_p of p. Using this POI-driven knowledge, our scoring conveys the POI semantic information in both TF and IDF.

Specifically, we use three types of frequencies: check-in frequency (*pf*), user frequency (*uf*), and POI frequency (*rf*). Given a tag w and a POI p, $pf(I_p, w)$ is the number of check-ins that have w in I_p. It is reasonable that a tag is likely to be one of the attribute tags as more check-ins of the POI have the tag. However, some users have the same tags in different check-ins causing overestimation of *pf*. Similarly, $uf(I_p, w)$ is the number of users that assign w in I_p. *uf* can control overestimated *pf*. However, we need to filter common tags like 'Travel', which also have high *pf* and *uf*. Given a tag w and a set p of all POIs, $rf(L, w)$ is the number of POIs $p \in L$ having w in I_p. Consider the rf distribution on the overall tag set. The head may contain tags that would be too generic attributes for all POIs, while tags in the tail (i.e., $rf = 1$) are likely not to be attribute terms. With these three types of frequencies, we define an attribute (AT) score of a tag w as:

$$AT(w) \propto \max_{p \in L} \frac{pf(I_p, w) \cdot uf(I_p, t)}{rf(L, w)} \tag{3}$$

We consider a tag w as an attribute keyword if $AT(w)$ is greater than a pre-defined threshold and $rf(L, w) > 1$.

B. Pattern Discovery Methods

To achieve the "Where, When, Who" consideration issue of user demands, the pattern discovery and scoring module defines the ranking mechanism for each POI with global attractiveness, proper visiting time and geo-social influence.

1) Determining the attractiveness scores of POIs: Below, we briefly introduce how to determine the attractiveness scores for POIs by mining the transition pattern.

Given a set of trajectories recorded as a series of check-in points, each check-in point represents a POI, which is denoted by $(latitude, longitude)$.

To measure the attractiveness of POIs, we intend to explore the sequential relationships hidden in the trajectory records. A region transition graph is built based on the transition probabilities to derive the traversal relationship. The example in Figure 1 can be revised to a region transition graph shown as Figure 3. The edges indicate the transition relationships among POIs. The weight of each edge is derived by aggregating the transition relationship from a set of trajectories. With the region transition graph, we borrow the concept in PATS [9] to adopt the Markov model to assign an attractiveness

[1] https://msdn.microsoft.com/library/ff701733.aspx

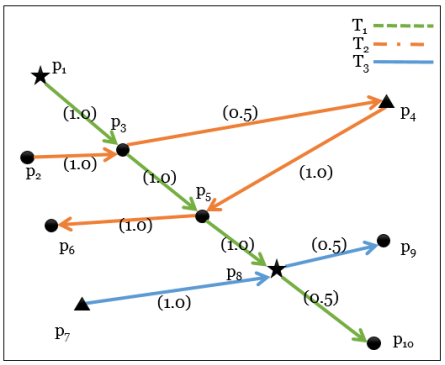


Fig. 3. An example of a region transition graph

score to each POI and denote the attractiveness score of POI p_i represented as $AS(p_i)$. According to PATS [9], the attractiveness of a trajectory can be inferred by a sequence of POIs and the concept of random walks.

2) Proper visiting time decision: As claimed in [10][2], the pleasure of visiting the POIs along a route is closely related to the arrival time. Some places have a wider range of visiting time period while others are constrained to certain particular time slots. For example, most people do not go to night markets in the morning, but rather arrive in the evening after the market opens. Thus, we extract time information from the timestamp of check-ins and mine the proper visiting time period of each POI. As shown in Figure 4, each POI has a different time distribution curve. For example, the distribution of Shihlin Night Market in Figure 4(a) shows that most people have check-ins during the evening and we can infer that the proper visiting time of Shihlin Night Market is about 17:00 to 23:00. Another example in Figure 4(b) shows that most people visit Shilin Presidential Residence during the daytime so the proper visiting time may be 7:00 to 17:00. When the distribution curve approaches a horizontal line, it indicates that the POI is suitable to be visited at any time. On the contrary, when the distribution curve has a peak segment, it indicates that the POI may be popular at a special time and proper to visit at that time. The same concept can be used in the case of seasons, weekends and weekdays.

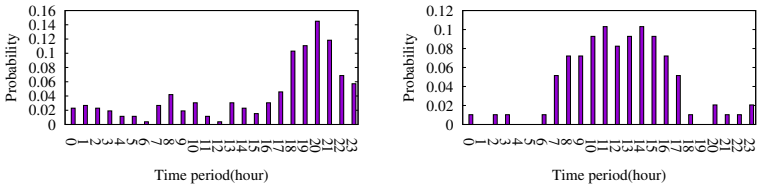
Here we separate one day into 24 time intervals. We first define the probability of a POI p be visited at time t as $N(p,t)/N_{total}(p)$, where $N(p,t)$ is the number of people who have check-ins in p at time t, and $N_{total}(p)$ is the total number of people who visit p. Due to the concern of data sparsity, we apply the 2-dimensional Gaussian $\mathcal{N}_t(\mu, \sigma^2)$ to fit the distribution of visiting time, which is similar to the idea of the temporal score in Section III-A2.

$$P(p,t) = \frac{1}{\sqrt{2\pi}\sigma} exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \quad (4)$$

After defining the visiting probability, we define the visiting time score $VS(p,t)$ for POI p at time t between 0 and 1 as:

$$VS(p,t) = \frac{P(p,t)}{max_{t\in 0\to 23}(P(p),t)} \quad (5)$$

where $max_{t\in 0\to 23}(P(p),t)$ is the maximum probability of visiting p in one day.



(a) Shihlin Night Market (b) Shihlin Residence

Fig. 4. The distributions of the visiting probability at each time unit (hour) for (a)Shihlin Night Market, and (b)Shilin Presidential Residence from check-in records.

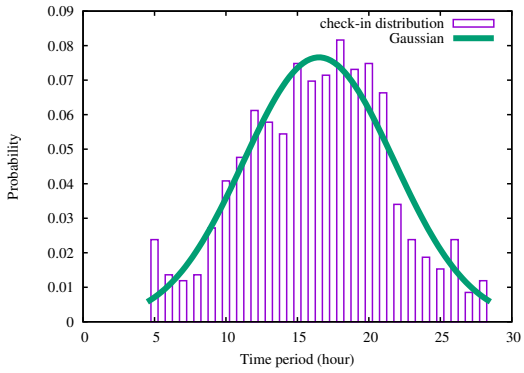
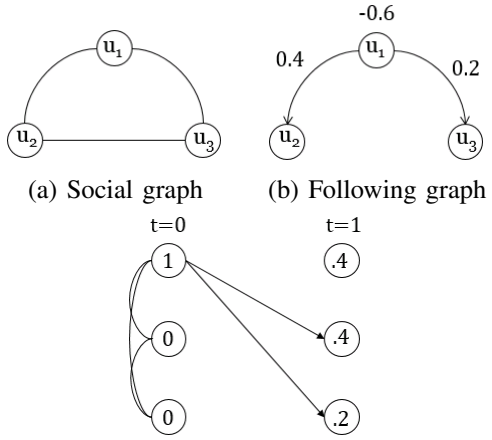


Fig. 5. An example of visiting time scoring. Two-dimensional Gaussian distribution is adopted to deal with inconsistent data due to data sparsity.

Now we give an example to illustrate how to compute VS. Given a visiting time Gaussian distribution of p in Figure 5, the maximum visiting time probability of p, i.e., $max(P(p))$ is 0.08 at the 19:00 time interval, the probability of the 10:00 interval is 0.04, and $VS(p,10) = 0.04/0.08 = 0.5$.

3) Geographical social influence: Another important feature in POI recommendation is the social influence of each user. The closer the relationship between two users, the more reliable the recommendation is. We focus on the condition that if user u_i is socially influenced by user u_j, u_i would visit a POI because of the recommendation from u_j. We can observe a relation that u_i checks in at the same POI p after u_j shared her/his check-in. [8] proposed a Social Influence Recommender (SIR) framework to explore the influential users in an LBSN. The main issue is to capture the interaction among the social network, physical locations and time effect by defining the geo-social following relation among users. Then a diffusion model is utilized to stimulate the influence spreading process. The example in Figure 1 can also be revised to Figure 6 to measure the geo-social influence among users.

The social influence can propagate among the users within the *following graph*. Formally, the amount of influence transited from u_i to u_j is defined as $f_{ij} = \frac{p(u_i,u_j)}{n(u_i)}$. $p(u_i,u_j)$ denotes the probability of geo-social following relationships of u_i to u_j, and n_i denotes the total number of locations u_i has visited. [8] assumes that user $u_i \in V$ is only influenced by her/himself initially and the influence will be propagated to others in the *following graph* afterwards. During the propagation process, users receive stimulation from their neighbors. Finally, vector $s(t)$ denotes the proportion of the social influence score on users in V at time t, the change at



(a) Social graph (b) Following graph

(c) Influence propagation from t = 0 to t = 1

Fig. 6. An example of influence propagation among three users.

u_i between time $t + \Delta t$ is defined by the following equation using the diffusion model:

$$\frac{s(t+\Delta t) - s(t)}{\Delta t} = \alpha Inf s(t) \tag{6}$$

where α is the propagation coefficient and Inf is a matrix which defines the one-hop information diffusion process.

IV. Travel route exploration

In this section, we present the algorithm of trajectory reconstruction and how to apply Skyline query for our online recommendation system. Furthermore, we propose an approximate algorithm to speed up the real-time Skyline query. The *Travel Route Exploration* procedure is presented as Algorithm 1.

Algorithm 1: Travel routes exploration

Input: Specific user u, query range Q, a set of keywords K and threshold ϵ;
Output: Keyword-aware travel routes with diversity in goodness domains *KST*.

```
Initialize priority queues CR,KST;
Scan the database once to find all route histories covered by region Q;
/* Fetch POI scores and check keyword matching */
foreach route r found do
    r.kmatch ← 0;
    foreach POI p ∈ r do
        r.kmatch ← r.kmatch + KM(p,k);
    if r.kmatch > 0 then
        Push r into CR;
/* Candidate route generation, see Algorithm 2 */
Append Reconstruct(CR) to CR;
/* Skyline search */
foreach route r ∈ CR do
    foreach candidate route r_c ∈ KST do
        if r_c.score dominates r.score then
            r is discarded as not part of the skyline;
        if r.score dominates r_c.score then
            Pop r_c from KST;
        Push r into KST;
return KST.
```

A. Query Keyword Matching

To process the user queries, we first describe how to match query keywords with the characteristic scores assigned to tags. The user-specific keywords in the query reflect the individual's preferences regarding the trip, i.e., the user tends to choose a travel route that contains POIs closely related to the semantic meanings. In the offline model, we have built a tag corpus for POIs with characteristic scores and metadata. Also, relevant tags for each POI are weighted in the TFIDF manner. Given a keyword set K and arbitray POI p at query time, we define a keyword matching measure **KM** with the pre-computed information:

$$KM(p,K) = \sum_{k \in K} tfidf(k,p) \cdot (GS(k) + TS(k) + AT(k)) \tag{7}$$

where *tf* is the frequency of tag k in a POI and *idf* is the number of POIs with the tag k.

For example, consider that given the keyword set K = ["night" "ximending"], we then find the temporal score of "night"= 0.9 and the geo-specific score = 0.001; the temporal score of "ximending" = 0.5 and the geo-specific score = 0.95. On the other hand, in a POI "red house", the TFIDF score of night = 0.3 and the TFIDF score of ximending = 0.8. These scores of keyword set K can be aggregated for POI "red house" as score (0.3 × (0.9 + 0.001)) + (0.8 × (0.5 + 0.95)). For the route with multiple POIs, the score of each POI as computed above will be summed up. The higher the score, the more related the route is with the keyword. We filter out the routes with zero score, which means that those routes are not related to the user's preference.

B. Candidate Route Generation

We have proposed the method for matching raw texts to POI features in each travel route candidate to fulfill the requirement that recommended travel routes should connect to all or partial user-specific keywords. Notice that the trajectories found in the previous section are limited to the existing trajectories. However, the existing trajectories sometimes may not include all the query criteria, and may have bad connections to the query keywords. Thus, we propose the *Candidate Route Generation* algorithm to combine different trajectories to fit the query requirement.

The new candidate routes are constructed by combining the subsequences of trajectories. Here we introduce the pre-processing method first. We then utilize the pre-processing results to accelerate the proposed route reconstruction algorithm. Last, we design a Depth-first search-based procedure to generate possible routes.

Pre-processing: After the *Query Keyword Matching* check, all the input routes must have at least one user-specific keyword. With the information that a trajectory T_i consists a sequence of POIs, $\{p_1, p_2, ...p_n\}$. Then we use the data structure (head,tail) to reinterpret the trajectory for one-step transition, i.e., $\{p_1 \rightarrow p_2, p_2 \rightarrow p_3, ..., p_{n-1} \rightarrow p_n\}$. Two dictionaried lists $headSet$ and $tailSet$ are used to record the head and tail records respectively.

Combined points should be ordered by time: Obviously, it is intuitive to combine (p_i, p_j) and (p_k, p_l) if p_j and p_k are

Algorithm 2: Candidate Route Generation

Input: Route set R;
Output: Candidate route set CR.
1 Initialize a stack S, priority queue CR;
2 Split $r \in R$ into (head,tail) subsequences;
3 Reconstruct(headSet).
4 Procedure Reconstruct(Set):
5 **foreach** *(head,tail)* $\in$ *Set* **do**
6 endFlag = False;
7 **if** *S is empty or tail.time* $>$ *S.pop().time* **then**
8 Push head in S;
9 Push tail in S;
10 **else**
11 Push head in S;
12 endFlag = True;
13 **if** *endFlag is False* **then**
14 Reconstruct(tailSet)
15 Add S in CR;
16 Procedure End

the same location. Besides considering spatial distance, we also need to consider the visiting time order among combined points. Since *tail.time* must be larger than *head.time*, p_k*.time* should be larger than p_i*.time* in order to replace p_j by p_k.

DFS-based route enumeration: In order to generate all possible routes from original trajectories, we reconstruct new trajectories by linking the (head,tail) subsequences using combined point. This would be a depth-first search-based procedure. We consider all the POIs in the headSet as source, and explore as far as possible along each link before backtracking. Furthermore, it is relatively straightforward to consider the time-period query in the process of DFS. In Algorithm 2, we only need to add a more strict time-query limitation to line 7, e.g., the size of S should not exceed a *the number of POIs in a route* threshold.

For example, the three existing travel routes T_1, T_2 and T_3 from Figure 1 can be reinterpret into (head,tail) pairs, as shown in Table II. Then we have the $headSet$ $\{p_1$,p_2,p_3,p_4,p_5,p_7,$p_8\}$. Start from p_1, $\{p_1$ (10:00) $\rightarrow$ p_3 (12:00)$\}$ is found first. p_3 is the combined point to $\{p_3$ (12:30) $\rightarrow$ p_4 (17:00)$\}$ since the visiting time order is correct. Finally, a candidate route T'_4 is generated as $\{p_1$ (10:00) $\rightarrow$ p_3 (12:30) $\rightarrow$ p_4 (17:00) $\rightarrow$ p_5 (19:00) $\rightarrow$ p_6 (19:30)$\}$. Table III shows the result of candidate routes: T_1 - T_3 are original routes and T'_4 - T'_6 are three of the reconstructed routes.

TABLE II. RAW TRAJECTORY DATASET.

Tid	(head,tail) subsequence	
T_1	p_1 (10:00) $\rightarrow$ p_3 (12:00)	p_3 (12:00) $\rightarrow$ p_5 (15:30)
	p_5 (15:30) $\rightarrow$ p_8 (17:30)	p_8 (17:30) $\rightarrow$ p_{10}(19:00)
T_2	p_2 (10:30) $\rightarrow$ p_3 (12:30)	p_3 (12:30) $\rightarrow$ p_4 (17:00)
	p_4 (17:00) $\rightarrow$ p_5 (19:00)	p_5 (19:00) $\rightarrow$ p_6 (19:30)
T_3	p_7 (18:30) $\rightarrow$ p_8 (19:30)	p_8 (19:30) $\rightarrow$ p_9 (20:00)

TABLE III. SUBSET OF CANDIDATE ROUTES.

Tid	POI sequence
T_1	p_1 (10:00) $\rightarrow$ p_3 (12:00) $\rightarrow$ p_5 (15:30) $\rightarrow$ p_8 (17:30) $\rightarrow$ p_{10}(19:00)
T_2	p_2 (10:30) $\rightarrow$ p_3 (12:30) $\rightarrow$ p_4 (17:00) $\rightarrow$ p_5 (19:00) $\rightarrow$ p_6 (19:30)
T_3	p_7 (18:30) $\rightarrow$ p_8 (19:30) $\rightarrow$ p_9 (20:00)
T'_4	p_1 (10:00) $\rightarrow$ p_3 (12:30) $\rightarrow$ p_4 (17:00) $\rightarrow$ p_5 (19:00) $\rightarrow$ p_6 (19:30)
T'_5	p_1 (10:00) $\rightarrow$ p_3 (12:00) $\rightarrow$ p_5 (19:00) $\rightarrow$ p_6 (19:30)
T'_6	p_1 (10:00) $\rightarrow$ p_3 (12:00) $\rightarrow$ p_5 (15:30) $\rightarrow$ p_8 (19:30) $\rightarrow$ p_9 (20:00)

C. Skyline Travel Routes Search

Given a data set D, a Skyline is a subset of data which stands out among others and is of special interest to us in D. More formally, a Skyline is a subset of data in D which is not dominated by any others. Let a and b be data points in D, a dominates b if a is as good as or better than b in all dimensions and better in at least one dimension. Instead of using a traditional recommendation system considering a fixed weighting for a set of criteria and returning the top K trajectories with the highest score, Skyline query considers all possible weighting criteria that might offer an optimal result.

Unlike the top-K system where users are required to specify weighting for a set of criteria and retrieval size K, Skyline does not need to make such a difficult decision. To illustrate, we have given an example of Skyline in the introduction and described the three scoring criteria in previous sections. Now we give the definition of the travel route Skyline as follows:

Definition 1: (Skyline travel route): Given a trajectory set, we have already retrieved the scores of attractiveness, time, and geographical social influence scores in the previous sections. The results of the Skyline travel route are not dominated by any other trajectory. Consider the three dimensions previously mentioned, i.e., attractiveness, time, and geographical social influence score; trajectory T_i dominates trajectory T_j if and only if the score of T_i in any dimension is not less than the corresponding score of T_j, where i is not equal to j.

In other words, the user need not specify the weight between every criteria on first because travel route Skyline returns all the possible optimal results w.r.t. arbitrary weight. In our system, the user can choose the travel route considering the different weight in three dimensions: (i) how attractive this trajectory is, (ii) the proper visiting time of each POI in the travel sequence, and (iii) the social influence of the users who have visited the POI. Each trajectory is regarded as a three-dimensional data point and each dimension corresponds to one score. After computing the trajectory scores, we return the Skyline travel routes as the recommended travel routes.

D. Greedy Pruning

Recall Section IV-B, a brute-force approach to generate the candidate routes is to enumerate all subsequences. The time complexity of candidate route generation is $\mathcal{O}(n^l)$, where n is the number of original routes and l is the average length of the routes. Also, the amount of reconstructed routes is $\mathcal{O}(n^l)$ because the routes are generated during the traversal. Since we only require the Skyline results, the redundant time cost of computing the scores and searching for low-ranked routes is computationally prohibitive.

As discussed in Section III, there are three dimensions to be considered in a Skyline search. Generally, POIs with higher scores in each dimension have higher probability not to be dominated. We leverage a greedy pruning method to reduce the number of candidate routes. The idea is that we only use the top-$N\%$ ranked POIs to construct routes. We define the greedy score of POI element p as

$$GPscore(p) = p.patscore + p.timescore + p.socialscore \tag{8}$$

In particular, the greedy pruning should be applied before the candidate route generation, and then use the pruned results to reconstruct candidate routes.

Algorithm 3: Greedy Pruning Algorithm

Input: Route set R, pruning threshold $N\%$;
Output: Route set after greedy pruning *GPR*.

```
foreach route r found do
    r.gpscore ← 0;
    foreach POI p ∈ r do
        r.gpscore ← r.gpscore + GPscore(p);
Sort R with r.gpscore;
GPR ← top-N% of R;
return GPR.
```

The sorting takes $\mathcal{O}(n \log n)$ time and then the procedure of candidate route generation can be reduced to $\mathcal{O}(n^{N\% \times l})$. Moreover, the time cost on Skyline search decreases exponentially as the number of candidate routes decreases. So we get the approximate results efficiently.

E. System Implementation

We implemented the system on an x86_64 Linux server with 16 cores and 8 GB memory. All the scores mentioned in Section III are computed offline and stored in a PostgreSQL 9.3 database with GIS extension.

Assume the number of routes in the dataset is N, and the average length of the routes is l. The time complexity of our *Travel Route Exploration* algorithm depends on three parts: (i) scan the whole database to find the routes in the query range, (ii) generate candidate routes and (iii) calculate feature scores and run Skyline search on all generated candidate routes. First, the search for (i) takes $\mathcal{O}(N)$ and gets even faster since the R-tree based GIS index filters out non-candidate routes efficiently. For step (ii), we have proposed a approximate mechanism to reduce the computations. And for each candidate route, step (iii) computes the scores and compares the domination to other routes. The complexity is $\mathcal{O}(N^2 \times l)$. In the case of extensive routes returned from a large-scale query region, it leads to excessive computational time and is not applicable for an interactive online system. We optimize the implementation by parallelizing score comparison in step (iii), which involves independent computations of each route. See Section V-B for the optimized run time results.

V. Experiments

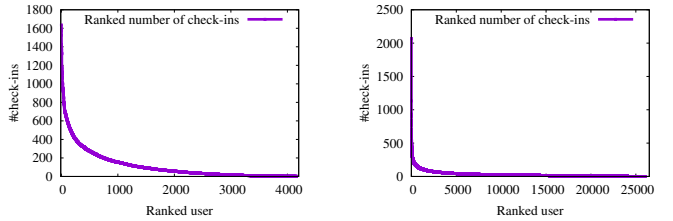
In this section, we empirically evaluate the effectiveness and efficiency of the proposed algorithms. First, we describe the baseline approaches and evaluation methodology of the experiments. We use two real-world LBSN datasets shown in Table IV. The *FB* dataset is collected by Facebook API[2]. We have taken 96 volunteers' Facebook accounts as user seeds (most of the users live in Taiwan) and crawled all their and their friends' location records (i.e., check-ins and geo-tagged photos) over the period of Jan. 2012 - Dec. 2014. *CA* is another Foursquare dataset with an undirected friendship network from [11]. All the experiments are conducted on an x86_64 Linux server with 16 cores and 8 GB memory.

[2]Facebook Developers. https://developers.facebook.com/

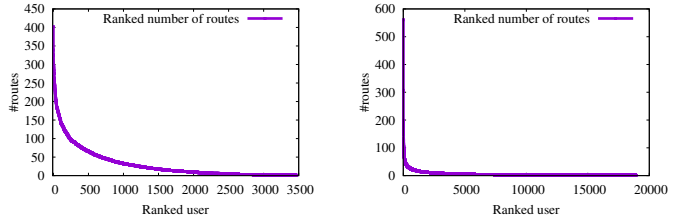
TABLE IV. Details of the LBSNs

	Property	Network	
		FB	**CA**
#records	check-in	869,317	483,813
#nodes	user	29,512	4,163
	POI	225,077	121,142
#edge	friend	39,513	32,512

To gain insights into the datasets, we plot both the number of check-ins and routes of each user of our datasets. As shown in Figure 7, the number of check-ins and routes for each user is highly skewed in both datasets. Moreover, all distributions have long tails. In particular, the top 10% ranked users in all datasets have nearly 60% of total check-ins and routes. This indicates that most of the users are quite inactive. The data sparsity issue may cause considerable bias on the result of inactive users. So we chose the top 10% of users, who were ranked by the travel route histories they have, as active users for testing.



(a) Check-in distribution of *CA* dataset (b) Check-in distribution of *FB* dataset



(c) Route distribution of *CA* dataset (d) Route distribution of *FB* dataset

Fig. 7. The number of check-ins and the number of routes for all uses in *CA* and *FB* dataset, respectively. The distribution shows a long tail extending in the negative direction.

A. Keyword matching accuracy

In this subsection, we evaluate the quality of the extracted keywords. Since our check-in datasets do not have sufficient text descriptions, i.e., tags, we collected an additional photo dataset consisting of 165,057 photos with 958,441 tags in the same local area. For that, the tags are regarded as input keyword. We ranked the tags by using scores in Section III-A and measured precision@K. Table V shows the precision of deciding the Geo-specific, Temporal, and Attribute keywords.[3] We can see that the precision is reasonably high and does not decrease much as K increases. Table VI shows the results for keyword extraction. Note that the keywords in italics are the Chinese keywords returned, which we translate for presentation. In the geo-specific dimension, 10 keywords referring

[3]For attribute extraction, we adopt [12] extracting probable attributes of all possible concepts. We can adopt 10 concepts aligned with POI categories, and Table VI illustrates attributes of 'Food' concept for restaurant POIs.

to certain places are highly ranked. For example, a keyword 'Longshan' represents 'Longshan Temple'. In the temporal dimension, there is no doubt that keywords such as 'Sunset', 'Sunrise', 'Lunch' and 'Night' are specific to a certain time interval. 'Dadaocheng' is ranked high as it is a place famous for its sunset. Also, 'Butterfly' and 'Fireworks' are strongly associated with day time and night time respectively. In the attribute dimension, keywords relevant to restaurant POIs are highly ranked.

TABLE V. PRECISION OF KEYWORD EXTRACTION

	P@10	P@20	P@40
Geo-specific keyword	1.000	1.000	0.975
Temporal keyword	0.900	0.700	0.720
Attribute keyword	1.000	0.850	0.775

TABLE VI. TOP-10 RESULTS OF KEYWORD EXTRACTION

	Keyword types		
	Geo-specific	Temporal	Attribute
1	Longshan	*Sunset*	Recipe
2	*Guanghua digital plaza*	Sunset	Soup
3	*Huashan creative park*	*Sunrise*	Store
4	*NTN univ.*	*Dadaocheng*	Oil
5	*Dadaocheng dock*	*Fireworks*	Sale
6	*Forty-four village*	Fireworks	Butter
7	*Taipei fine arts museum*	Butterfly	Sauce
8	*Three gorges street*	Boat	Bread
9	Ximending	Lunch	Chicken
10	*CKS memorial hall*	Night	Delivery

B. Efficiency

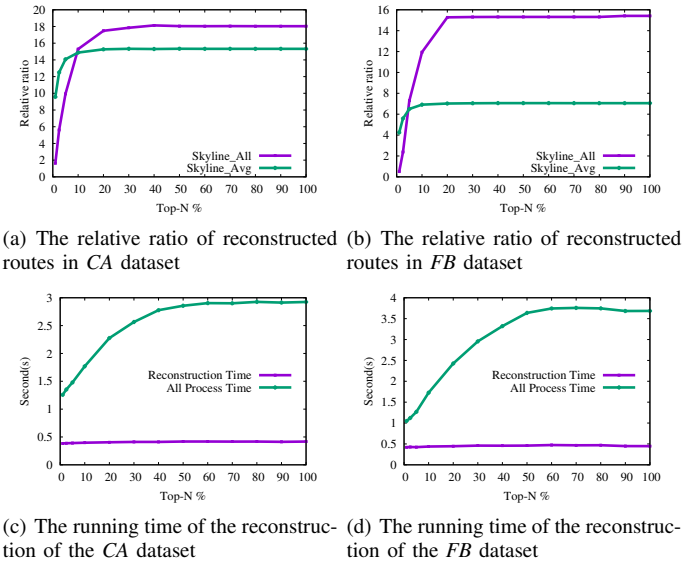
Table VII shows the online response time of *KSTR* in three main sub-procedures: (i) generate candidate routes (Reconstruct), (ii) compute the score of original/reconstructed routes (O_scoring+R_scoring), and (iii) Skyline search (Skyline). We synthesize 34,928 queries from testing users of *FB* dataset and 39,729 queries from *CA* dataset. The average response is 1.561708549 seconds. We can find that **Reconstruct** is the most time-consuming step. In Subsection V-B1, we observe the optimal N for approximate candidate route generation. The total running time under different scales is shown in Subsection V-B2.

TABLE VII. RUNNING TIME RATIO (SUB-PROCEDURE TIME COST / TOTAL TIME COST) OF EACH STEP.

	Reconstruct	**O_scoring**	**R_scoring**	**Skyline**
FB	0.265513757	0.160751505	0.040780763	0.099222254
CA	0.213114739	0.155153407	0.038816553	0.163912108

1) Tuning Approximate Parameters: First, we study the accuracy of the approximate routes reconstruction algorithm. We define the term "relative ratio" as the ratio of reconstructed routes to the Skyline searched results. By randomly choosing 1,000 routes in the testing set, we observe the optimal parameter N for selecting the top-$N\%$ ranked POIs to generate routes that controls the best trade off between effectiveness and running time. Figure 8 shows the average relative ratio of the 1,000 testing routes compared to the value of N. Note that the brute-force method is $N = 100$.

As shown in Figures 8(a) and 8(b), we can find that the relative ratio of both datasets converges rapidly as N increases. Moreover, although the running time of reconstruction is only slightly longer when $N = 100$, the running time of the whole



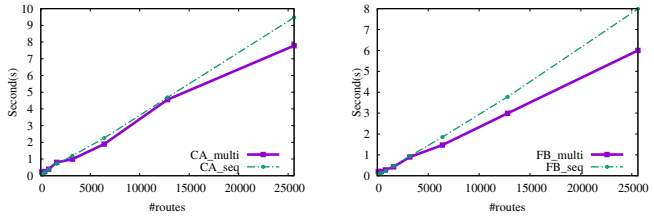
(a) The relative ratio of reconstructed routes in *CA* dataset (b) The relative ratio of reconstructed routes in *FB* dataset

(c) The running time of the reconstruction of the *CA* dataset (d) The running time of the reconstruction of the *FB* dataset

Fig. 8. The effectiveness of the candidate route generation of the *CA* and *FB* dataset, respectively under different top-$N\%$ of POI elements. The results converge as N increases.

procedure is obviously affected. The reason is that the number of generated routes increases exponentially w.r.t. the size of the POI elements. Moreover, the growth trend of route number levels off when $N > 50$. Therefore, we choose $N = 10$ in both datasets, which holds the accuracy and speed.

2) Scalability: The objective of this set of experiments is to study the scalability of the proposed algorithms with variation of the number of computations. We have made use of several methods to optimize the implementation of the online system. Figure 9 shows the total running time and the comparison of the sequential scoring and the multiprocess[4] scoring. In general cases, the number of route computations of a user query seldom exceeds 5,000 and the response time of the query takes no more than one second. Since the result is sufficiently fast, the multiprocess mechanism does not lead to evident improvement. On the other hand, in extreme cases with 26,000 route computations, using multiprocess reduces 25% of time cost.



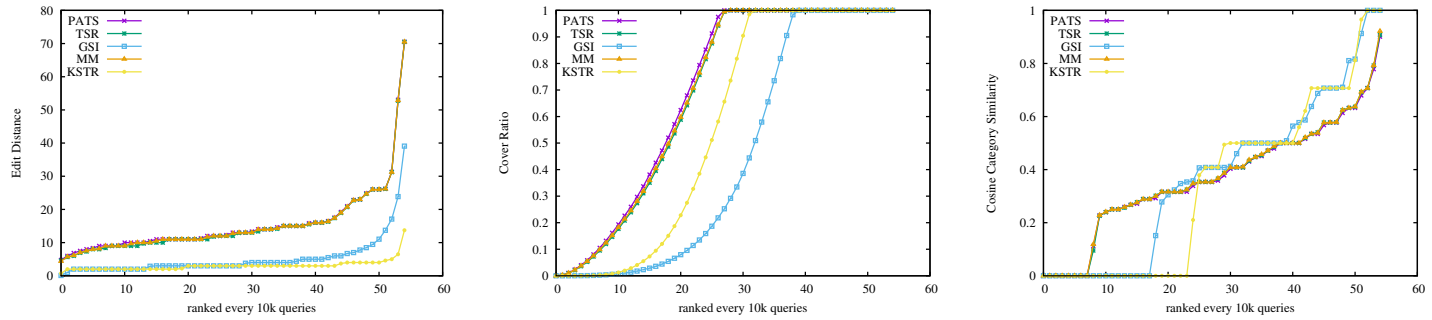
(a) The relative ratio of reconstructed routes in the *CA* dataset (b) The relative ratio of reconstructed routes in the *FB* dataset

Fig. 9. Runtime versus route number (computation size).

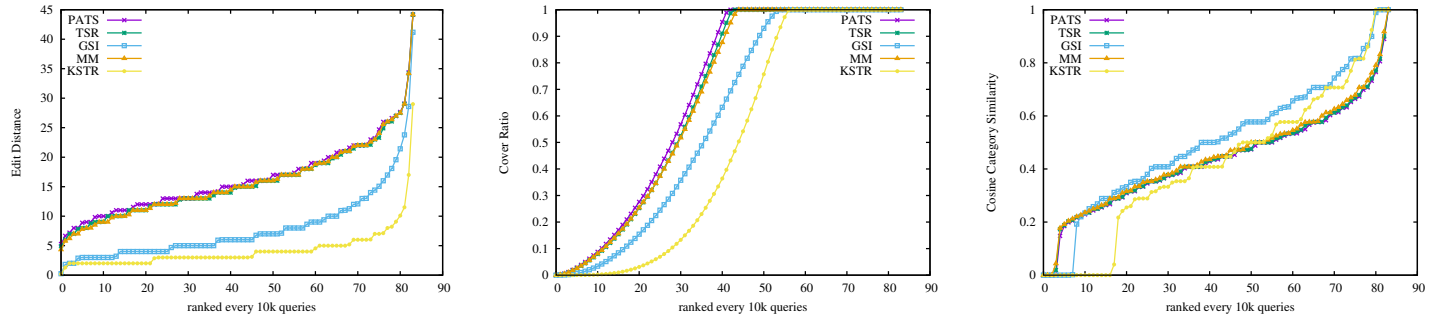
C. Evaluation of route prediction accuracy

In this experiment, we compared the following four baseline recommendation models to our keyword-aware skyline travel route (*KSTR*) model.

[4]Eight-cores multi-processing



(a) Average edit distance versus to recommended travel routes of *FB* dataset (b) Average region cover ratio versus to recommended travel routes of *FB* dataset (c) Average category similarity versus to recommended travel routes of *FB* dataset



(d) Average edit distance versus to recommended travel routes of *CA* dataset (e) Average region cover ratio versus to recommended travel routes of *CA* dataset (f) Average category similarity versus to recommended travel routes of *CA* dataset

Fig. 10. Average goodness accuracy of recommended travel route at different query region sizes. The yellow line represents our method and shows that KSTR has the good results over the three measurements.

Pattern aware trajectory search (PATS): Only consider the sum of the POI attractiveness score. Different to the Multinomial model, [9] consider the mobility transition among POI pairs.

Time-sensitive routes (TSR): Only consider the visiting time score of routes. The arrival time of the POIs in the recommendation best fits the extracted proper visiting time.

Geo-social influenced routes (GSI): Only consider the geo-social influence score of [8]. The route consists of POIs visited by geo-social influential users in social network.

Multinomial model (MM): This is the naive method that outputs the travel routes based on all three features. Routes are ranked by the summation of the feature scores.

Keyword-aware skyline route (KSTR): *KSTR* outputs diverse Skyline routes based on both POI and user factors.

Unfortunately, raw LBSN data provide no ground truth to verify the acceptance of the recommended travel route suggestions. Therefore, we studied the "appropriateness" of the recommended travel routes as a route prediction progress under different spare time conditions. We used the data shown in Table IV for training and testing the model. For each dataset, the test data were created by collecting the last travel sequence of the top-10% of users (ranked by route count) in the most recent 30% time periods. The training dataset consisted of the set of travel sequences excluding the testing data part. To be exact, the number of training data (the number of test data) used in this experiment is slightly larger than the number of testing data since users with multiple travel sequences only keep the last sequence.

1) Comparison of route prediction accuracy: We measured the difference between the generated routes and each test sequence. Three goodness functions are applied as the evaluation metrics.

Edit distance: The edit distance measures the distance between two sequences in terms of the minimum number of edit operations required to transform one sequence into the other [13]. The allowable edit operations are: insert into a sequence, delete from a sequence, and replace one landmark with another.

Geographical region cover ratio: The test route and recommended route both can be bounded by a geographical box. The ratio of the overlapped region to the testing route region.

Category similarity: To consider the closeness of user interest, we compute the cosine similarity of the categories between two routes, which is $\#\ of\ overlapped\ category\ /\ \sqrt{\#\ of\ category1 \cdot \#\ of\ category2}$.

We compared our *KSTR* model with the other models: multinomial model, pattern aware trajectory search, time-sensitive and geo-social influenced routes. Figure 10 shows the performance of each model among the three measures. We can find that the proposed *KSTR* model offers the lowest edit distance, which represents the highest prediction accuracy. On the other hand, considering the measure of region cover ratio and category similarity, the multinomial model has better performance than ours. The results show that the proposed *KSTR* is effective and beats other baselines and state-of-the-art methods in terms of route prediction accuracy.

D. Examples of Route Recommendation:

As a result, we implemented our proposed methods and constructed a web-based travel route recommender system. One example of recommended routes by the system is show.

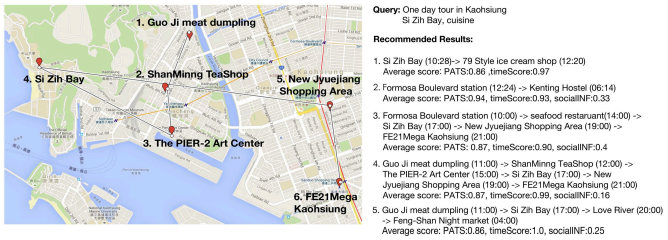


Fig. 11. Examples of KSTR routes in Kaohsiung.

Figure 11 shows the recommendation examples when the user gives a geographical region of Kaohsiung on the map interface and specifies a one day tour and keywords ["Si Zih Bay" "Cuisine"]. The system provides five different trip plans. Note that, unlike ranking, the ordering of skyline results does not reflect relevance. *Route #4* is visualized on the map interface and each icon on the map represents a POI. Besides the specific region "Si Zhi Bay", "Guo Ji meat dumpling" and "ShanMinng TeaShop" are perfectly associated with the keyword "Cuisine". And we can see the route with the best visiting time suggestions.

VI. Conclusion

In this paper, we study the travel route recommendation problem. We have developed a *KSTR* framework to suggest travel trajectories with a specific range and a set of user preference keywords. These travel trajectories are related to all or partial user preference keywords, and are recommended based on (i) the attractiveness of the POIs it passed, (ii) visiting the POIs at their corresponding proper arrival time, and (iii) the route generated by influential users. We propose a novel keyword extraction module to identify the semantic meaning and match measurement of routes, and design a route reconstruction algorithm to aggregate trajectory segments into travel routes in accordance with query range and time period. We leverage score functions for the three aforementioned features and adapt Skyline search for travel route recommendation instead of the traditional top-K system. The experimental results demonstrate that *KSTR* is able to retrieve travel routes that are interesting for users and outperforms baseline algorithms in terms of effectiveness and efficiency.

Acknowledgment

Wen and Peng were supported in part by MOST Taiwan, Project No. 102-2221-E-009 -171 -MY3 and 104-2221-E-009 -138 -MY2, by Academic Sinica Theme project, Project No. AS-102-TPA06. Hwang and Yeo were supported by ICT R&D program of MSIP/IITP [B0101-15-0307] and Microsoft Research.

References

[1] Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie, "Searching trajectories by locations: an efficiency study," in *SIGMOD*, 2010, pp. 255–266.

[2] H.-P. Hsieh and C.-T. Li, "Mining and planning time-aware routes from check-in data," in *CIKM*, 2014, pp. 481–490.

[3] V. S. Tseng, E. H.-C. Lu, and C.-H. Huang, "Mining temporal mobile sequential patterns in location-based service environments," in *Proceedings of the 13th International Conference on Parallel and Distributed Systems (ICPADS)*, vol. 2, 2007, pp. 1–8.

[4] W. T. Hsu, Y. T. Wen, L. Y. Wei, and W. C. Peng, "Skyline travel routes: Exploring skyline for trip planning," in *MDM*, vol. 2, 2014, pp. 31–36.

[5] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories for mobile users," in *WWW*, 2009, pp. 791–800.

[6] Q. Yuan, G. Cong, and A. Sun, "Graph-based point-of-interest recommendation with geographical and temporal influences," in *CIKM*, 2014, pp. 659–668.

[7] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *SIGIR*, 2011, pp. 325–334.

[8] Y.-T. Wen, P.-R. Lei, W.-C. Peng, and X.-F. Zhou, "Exploring social influence on location-based social networks," in *ICDM*, 2014, pp. 1043–1048.

[9] L.-Y. Wei, W.-C. Peng, B.-C. Chen, and T.-W. Lin, "Pats: A framework of pattern-aware trajectory search," in *MDM*, 2010, pp. 372–377.

[10] H.-P. Hsieh, C.-T. Li, and S.-D. Lin, "Exploiting large-scale check-in data to recommend time-sensitive routes," in *UrbComp*, 2012, pp. 55–62.

[11] H. Gao, J. Tang, and H. Liu, "Exploring social-historical ties on location-based social networks." in *ICWSM*, 2012.

[12] T. Lee, Z. Wang, H. Wang, and S.-w. Hwang, "Attribute extraction and scoring: A probabilistic approach," in *ICDE*, 2013, pp. 194–205.

[13] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotags in photo sharing sites," in *CIKM*, 2010, pp. 579–588.

[14] H. Wang, Z. Li, and W.-C. Lee, "Pgt: Measuring mobility relationship using personal, global and temporal factors," in *ICDM*, 2014, pp. 570–579.

[15] M. Ye, X. Liu, and W.-C. Lee, "Exploring social influence for recommendation: A generative model approach," in *SIGIR*, 2012, pp. 671–680.

[16] A. Sadilek, H. Kautz, and J. P. Bigham, "Finding your friends and following them to where you are," in *WSDM*, 2012, pp. 723–732.

[17] X. Cao, G. Cong, and C. S. Jensen, "Mining significant semantic locations from gps data," *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1009–1020, 2010.

[18] K. Zheng, S. Shang, N. J. Yuan, Y. Yang, and U. Computing, "Towards efficient search for activity trajectories," in *ICDE*, 2013.

[19] Z. Yin, L. Cao, J. Han, J. Luo, and T. S. Huang, "Diversified trajectory pattern ranking in geo-tagged social media," in *SDM*, 2011, pp. 980–991.

[20] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*, 2010, pp. 811–820.

[21] X. Lu, C. Wang, J.-M. Yang, Y. Pang, and L. Zhang, "Photo2trip: generating travel routes from geo-tagged photos for trip planning," in *MM*, 2010, pp. 143–152.

[22] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *SIGMOD*, 2003, pp. 467–478.

[23] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, "An energy-efficient mobile recommender system," in *KDD*, 2010, pp. 899–908.

[24] Y. Arase, X. Xie, T. Hara, and S. Nishio, "Mining people's trips from large scale geo-tagged photos," in *MM*, 2010, pp. 133–142.

[25] X. Cao, L. Chen, G. Cong, and X. Xiao, "Keyword-aware optimal route search," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1136–1147, 2012.

[26] B. Zheng, N. J. Yuan, K. Zheng, X. Xie, S. Sadiq, and X. Zhou, "Approximate keyword search in semantic trajectory database," in *ICDE*, 2015, pp. 975–986.

[27] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *KDD*, 2007, pp. 330–339.