A Travel Route Generation and Exploration Recommendation System on Social Media

Mr. Katkam Anurag, Mr. Vishal Korade, Ms. Shatavari Raut, Mr. Gaurav Kulkarni,
Prof. Navnath Bagal
Computer Department, TSSM's, PVPIT, Bavdhan Pune, Country - India - 411041.

Abstract - With the popularity of social media (e.g., Facebook and Flicker), users can easily share their check-in records and photos during their trips. In view of the huge number of user historical mobility records in social media, we aim to discover travel experiences to facilitate trip planning. When planning a trip, users always have specific preferences regarding their trips. Instead of restricting users to limited query options such as locations, activities or time periods, we consider arbitrary text descriptions as keywords about personalized requirements. Moreover, a diverse and representative set of recommended travel routes is needed. Prior works have elaborated on mining and ranking existing routes from check-in data. To meet the need for automatic trip organization, we claim that more features of Places of Interest (POIs) should be extracted. Therefore, in this paper, we propose an efficient Keyword-aware Representative Travel Route framework that uses knowledge extraction from users' historical mobility records and social interactions. Explicitly, we have designed a keyword extraction module to classify the POI-related tags, for effective matching with query keywords. We have further designed a route reconstruction algorithm to construct route candidates that fulfill the requirements. To provide befitting query results, we explore Representative Skyline concepts, that is, the Skyline routes which best describe the trade-offs among different POI features. To evaluate the effectiveness and efficiency of the proposed algorithms, we have conducted extensive experiments on real location-based social network datasets, and the experiment results show that our methods do indeed demonstrate good performance compared to state-of-the-art works.

Keywords - Location-based Social Network, Text Mining, Travel Route Recommendation.

I. INTRODUCTION

Location-based social network (LBSN) 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 route with some photos and tag information. As a result, a massive number of routes 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 [5] [11] [22] [13] [40] 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 routes simply by the popularity or the number of uploads of routes. For such ranking, the existing works [37] [32] [29] derive a scoring function, where each route 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 routes. Recently, [28] aimed to retrieve a greater diversity of routes based on the travel factors considered. As high scoring routes are often too similar to each other, this work considers the diversity of results by exploiting Skyline query.

In this paper, we develop a Keyword-aware Representative Travel Route (KRTR) framework to retrieve several recommended routes where keyword means the personalized requirements that users have for the trip. The route dataset could be built from the collection of low-sampling check-in records.

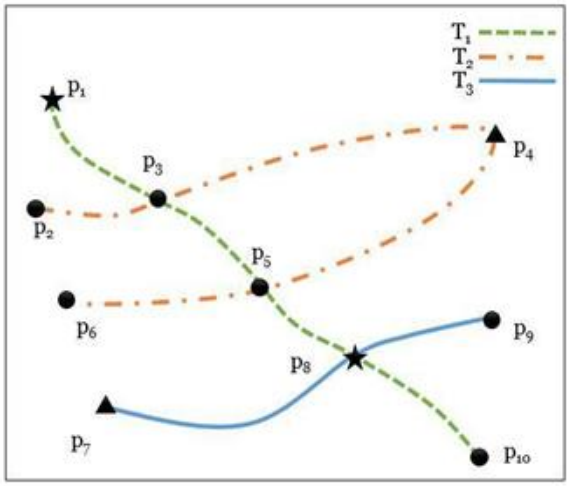


Figure. 1. Keyword-aware travel routes query running example.

Definition 1. (Travel route): Given a set of check -in points recorded as a series of travel routes, each check-in point represents a POI p and the user's checked-in time t. The check-in records were grouped by individual users and ordered by the creation time. Each user could have a list of travel routes fT g = fT0; T1; :::g, where T0 = (p0; t 0); (p1; t1); :::; (pi; ti), T1 = (pi+1; ti+1); (pi+2; ti+2); ::: and ti+1

ti is greater than a route-split threshold. We set the route-split threshold to one day in this paper.

Table 1: Example of route dataset

Tid	Uid	Pid	keyword	time	POI score vector
1	1	1	Opera House	10:00	(0.04, 0.2)
T_1	u_1	p_3	Bar	12:00	(0.25, 0.2)
1	1	5	Bar	15:30	(0.2, 0.8)
T_1	u_1	P_8	Opera House	17:30	(0.04, 0.3)
1	1	10	Bar	19:00	(0.04, 0.2)
T_2	u_2	p_2	Bar	10:30	(0.02, 0.2)
2	2	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)

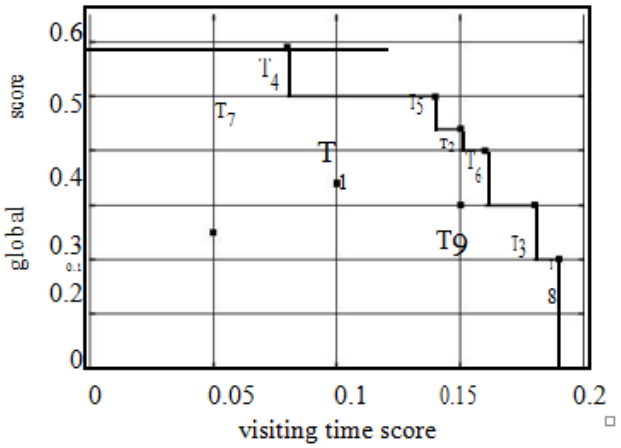


Figure. 2. An extended example of skyline travel routes built by Table 1.

Consider the example illustrated in Figure 1, the related route information of which is stored in Table 1. 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 represents 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 that knowing semantics is important, as some query keywords do not need to be matched in the POI keyword. For example, p9, 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, p8 matches the requirement. As a result, T3 matches all the requirements, which could not be supported by existing simple keyword-based matches. In this example, the keyword “Sunset” can be easily matched. Although the other two words are not stored in the database, we want to correspond them to Drinking whisky at a bar and Opera House in Sydney Cove. Finally, T 3 matches all the requirements. Meanwhile, there is still a possibility that no existing route is in accordance with the query keywords. For this challenge, we propose a candidate route generation algorithm to increase the number of routes. For instance, a travel sequence T 0 = fp1 ! p3 ! p4 ! p5 ! p8 ! p9g, which is aggregated from the route segments of T1 to T3, also matches all the keywords specified.

Additionally, we have mentioned that the final results may have similar characteristics and be monotonous due to the fact that all of the factors are aggregated into one score for each travel route. Consequently, the system will retrieve the top-k routes with the highest score as the results. Users may not understand the characteristic of these routes through the final single score (e.g., Which one has the most interesting landmarks? Which one is well-connected to the place I want to go?) so it may be hard to choose a route from the final results. Furthermore, users need to pre-define the weight for each factor, although it is hard to select a 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 borrowed the concept of Distance-based Representative Skyline [21] to retrieve travel routes. Distance-based Representative Skyline search on the travel routes also includes a small number k of skyline routes that best describe the full optimal (Skyline) results in terms of the features derived. 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 T1, T2 and T3, we get the final score values (0.1, 0.34), (0.15, 0.44), and (0.18, 0.3) respectively. For example, with k = 3, the skyline points in Figure 2 can be divided into three subsets fT4g, fT2,T5,T6g and fT 3,T8g. Our representative skyline travel route solution will report fT2; T 3; T4g.

This paper builds on and significantly improves the KSTR framework [28] of recommending a diverse set of travel routes based on several score features mined from social media. KSTR then constructs travel routes from different route segments. Specifically, we extend KSTR to consider representative and approximate results under an optional k limit in Section 5. Additionally, resources including passive check-ins such as GPS-tagged photos are discussed in Section 6. This addition would enable KRTR to consider a larger input including active and passive check-ins with high efficiency and scalability.

The contributions of this paper are summarized as follows:

We propose a KRTR framework in which users are able to issue a set of keywords and a query region, and for which query results contain diverse trip routes. Check-in information is mined from passive check-ins to enrich the input data. GPS-tagged photos are larger in scale than foursquare check-ins. This mining thus improves the coverage of the input data.

We propose a route reconstruction method to partition routes into segments by considering spatial and temporal features.

Representative Skyline query for travel route search is adopted to combine the multi-dimensional measurements of routes, which increases the diversity of the recommended results. Moreover, a greedy method is designed for the efficiency of the online application.

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

The rest of the paper is organized as follows. Section 2 presents the overview of the KRTR framework. Section 3 describes the feature scoring algorithms and how to extend KSTR to mine from both active and passive check-ins. In Section 4, we provide a travel routes exploration module of KRTR. The experiment results of the proposed methods are presented in Section 5. Section 6 summaries the related work. Finally, Section 7 concludes this paper.

II. FRAMEWORK OVERVIEW

In this section, the proposed framework KSTR is presented. KSTR is comprised of two modules: the offline pattern discovery and scoring module and the online travel routes exploration module. The notations used throughout the paper are summarized below in Table 2.

Table 2 Symbols and notations

Notation	Definition
p	location as Point-of-Interest (POI)
t	low-sampling route as travel sequence
w	tag that describes a POI
n	the number of routes in the dataset
K	a set of query keywords
GS(w)	geo-specificity score of a tag w
T S(w)	temporal-specificity score of a tag w
AT (w)	attribute score of a tag w
D	a set of d-dimensional featured routes
m	the number of routes in D
S	the full skyline of D
k	maximum number of the returned travel routes
R	the returned representative skyline travel routes

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 characteristics. Furthermore, we derive the feature scores of the POIs and generate proper candidate travel routes.

Online travel routes exploration module: In this module, we aim to provide an interface for users to specify query ranges and preference-related keywords. Once the system receives a specified range and time, the online module will retrieve those travel routes that overlap the query range and the stay time period. Then, it will compute a matched score of how well the travel route is connected to the keywords. Consequently, the online module returns the k most representative routes considering the aforementioned feature scores to the users.

III. PATTERN DISCOVERY

This section describes an offline process of pattern discovery from trajectory histories, which includes (1) the scoring mechanism for keywords and POIs; (2) a review of feature scoring methods that quantify the goodness of the routes; and (3) the candidate route generation algorithm.

3.1 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.

3.1.1 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 to get their representative locations and addresses. For a tag w, using GDA, we set GDA(w) as 1 if its location (latitude; longitude) 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 GeoV ar(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:

$$GS(w) / GDA(w) \exp(GeoV ar(w)) \quad (1)$$

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

3.1.2 Temporal keywords - Some tags are specific to a time interval, which represents its temporal nature. To quantify the temporal-specificity of a tag, time distribution on a tag rates the identified temporal-terms. Using the 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 TimeV ar(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:

$$T S(w) / \exp(T imeV ar(w)) \quad (2)$$

We consider a tag w as a temporal keyword if T S(w) is greater than a pre-defined threshold. Then, given a temporal keyword w, we generate a one-dimensional Gaussian Nt(; 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 deviations from.

3.1.3 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 Ip 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(Ip; w) is the number of check-ins that have w in Ip. 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(Ip; w) is the number of users that assign w in Ip. 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 L of all POIs, rf(L; w) is the number of POIs p 2 L having w in Ip. Consider the rf distribution of 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:

$$AT(w) \underline{\max pf(Ip; w)\, uf(Ip; t)} \quad (3)$$
$$/\ p\ 2L\ rf(L; w)$$

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

3.2 Passive check-ins

In previous sections, we worked with check-ins generated by users manually recording their whereabouts, such as foursquare check-ins of visiting Taipei 101. However, some such whereabouts are only passively recorded, such as photos of Taipei 101. Considering that six billion public photos have been uploaded in Facebook and more than 3% of photos have geo data 2 3, the volume of geo-tagged photos is 2.5 times larger than that of active check-ins. In addition, they capture locations that cannot be covered by active check-ins, such as new restaurants yet to be registered at Foursquare DB. We study how such passive check-ins can be leveraged, by extending our framework KSTR [28].

Our goal is to extract a check-in triple, hwho, where, wheni from a Flickr photo. As who and when are often clear from the user ID and the timestamp, we focus on extracting where based on the location and tags of the photo. However, this task is non-trivial, as users describe the same POI, such as Taipei 101, using many different names. For example, photo uploaders prefer to use various synonymous tags to refer to the same POI, which do not necessarily match with the official POI name. Besides, not all people assign tags referring to POIs taken in photos. To overcome the informal nature of photo tagging, we present a two-phase method for extracting check-ins from Flickr photos. The first phase identifies synonymous tags of an official POI name by exploiting characteristics of POIs. Considering the synonyms found, the second phase harvests virtual check-ins by propagat-ing POI-relevance scores through duplicate/near-duplicate photos.

3.2.1 Phase I: Synonym-based Check-in Extraction - The first phase is extracting a set Np of semantically equivalent terms (i.e., synonyms) of an official name np of a POI p. To be specific regarding POIs, considering photo tags as synonym candidates, we leverage rich signals associated between POIs and photos. Specifically, to extract tags synonymous with np, we quantify the location signals of a candidate tag t and image signals between np and t obtained from an estimated photo set Ip. Toward this goal of mining many synonyms, we have devised a scoring function which gives a high score for a and keyword that is likely to be the name. To devise such a scoring function, we adopt KSTR metrics.

Geo-specificity GS (Eq. 1): Some name tags are specific to the given location, which represents its spatial nature leading to a higher likelihood that it refers to a POI.

POI-specificity AT (Eq. 3): Among geo-specific keywords, we consider names frequently associated with the given POI (TF), which are not so much associated with other POIs (IDF).

Considering both scores in Eq. 1 and 3, we compute a synonym score of a tag w of a POI p as:

$$Synonym(p; w) = GS(w) + (1)\ AT(w);\ w\ 2\ Wp$$

where 0 1. is a weight parameter between GS(w) and AT (w) and Wp is a set of tags co-occurring with a tag np. Finally, if Synonym(p; w) is greater than a synonym threshold , we add w to Np.

3.2.2 Phase II: Collective Check-in Extraction - Once the synonym set Np is found, we can find a set of matching clusters among duplicate/near-duplicate [41] photo clusters Cp. We find c 2 Cp such that 9h 2 c \ Hp. Given c 2 Cp, we compute P (Npjc), which represents how relevant a cluster c is to a POI characterized by Np. The photo set Ip is then approximated as an aggregation of the clusters, i.e., [c, such that P (Npjc) is greater than a linking threshold. However, poor clusters in emerging nature cannot have sufficient tags and so this linking rule is still too strict to achieve high recall in finding photos.

To loosen it, a cluster cu can be matched with a POI p if a cluster cj is annotated with N p and we can answer the question "Do two clusters cu and cj refer to the same POI?". For that, we adopt a Bayesian approach to derive such a relationship by POI-semantic similarity between the clusters. Specifically, P (Npjc) is obtained from a pseudo-generative model using Bayes' Rule. Given two clusters cj and cu, we combine the two clusters. P (cj; c u) representing the tag similarity of two clusters and P (Nejc j) representing the reliability of cj for representing a POI e as follows:

$$P(pjcu)\ P(Npjcu) = \sum_{cj\ 2Cp} P(cjjcu)P(Npjcj)$$

Strictly speaking, neither the generative process from cu to cj nor the generative model from cj to Np are known or

defined precisely; hence the above conditional probabilities cannot be known exactly. However, we are not interested in probabilities per-se, but rather in probability values as indicators used eventually for linking the decision with.

For this reason, we can use proxy quantities – respectively a cluster-to-cluster similarity and a POI-to-cluster relevance which are presented as below.

The term P (cjjcu) represents the probability of generating the contents of a cluster cj from the contents of another cluster cu. As the contents, we consider textual knowledge, i.e., tags semantically-enriched by duplicate/near-duplicate photo clustering. We thus identify the tag frequency vector of each cluster and check whether two clusters share many co-occurring tags. Specifically, to estimate P (cjjcu), the cosine similarity of the cluster pair is calculated based on the Bag-of-Words model:

$$Sim(c_j; c_u) = \frac{T_{c_j} \; T_{c_u}}{T \qquad T}$$

where Tc is a frequency vector of tags annotated in a photo cluster c. All tags are weighted using term frequency inverted document frequency (TFIDF) intuition, abstracting a photo cluster cu as a document. The detailed formula will be discussed later. Now a proxy of the probability P (cjjcu) can be obtained by normalizing the content similarity between cu and cj according to the total similarity between cu and Cp:

$$P(c_j|c_u) = \frac{Sim(c_j; c_u)}{\sum_{c_k \in 2^C} Sim(c_k; c_u)} \qquad (4)$$

The term P (Npjcj) can be interpreted as an indicator of how reliably a photo cluster represents a POI. We directly derive the proxy value for this term using a simple frequency-based approach as follows:

$$P(N_p|c_j) = \frac{jN_e \setminus T_{cj}\, j}{\sum_{q} jN_p{}^{0} \setminus T_c\, j}$$

T annotated in a photo cluster c. where cj is a set of tags P_j

3.3 Feature Scoring Methods

With a set of travel route records, feature scoring should be considered to find proper recommendations. In this paper, we also 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". 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 [28]. From the viewpoint of the POI, we store the attractiveness score and the visiting time information in the POI score vector. On the other hand, from the viewpoint of the user, we also consider a score to quantify an individual's influence in recommendation.

3.4 Candidate Route Generation

In the previous sections, we have proposed the methods for matching raw texts to POI features and mining preference patterns in existing travel routes. However, the route dataset 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 routes to increase the amount and diversity. 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 - With the information that a trajectory Ti consists of sequence of POIs, f p1 ; p2; :::pn g , we use the data structure (head, tail) to reinterpret the trajectory for one-step transition, i.e., fp1 ! p2; p2 ! p3; :::; pn 1 ! png. 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 (pi; pj) and (pk; pl) if pj and pk are 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, pk.time should be larger than pi.time in order to replace pj with pk.

DFS-based route enumeration - In order to generate all possible routes from their original trajectories, we reconstruct new trajectories by linking the (head,tail) subsequences using combined points. This would be a depth first search-based procedure. We consider all the POIs in the headSet as the source, and explore as far as possible along each link before backtracking.

Algorithm 1: Candidate Route Generation

```
Input: Raw trajectory set T;
Output: New candidate trajectory set Tc.
1  Initialize a stack S;
2  Split each route r 2 T into (head,tail) subsequences;
3  Reconstruct (headSet).
4  Procedure Reconstruct(Set):
5  foreach (head,tail) 2 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     Insert S in Tc;
16 Procedure End
```

For example, the three existing travel routes T1, T2 and T3 from Figure 1 can be reinterpreted as (head, tail) pairs, as shown in Table 3. Then we have the headset fp1, p2,p3,p4,p5,p7,p8g. Starting from p1, fp1 (10:00) ! p3

(12:00)g is found first. p3 is the combined point to fp3 (12:30) ! p4 (17:00)g since the visiting time order is correct. Finally, a candidate route T40 is generated as fp1 (10:00) ! p3 (12:30) ! p4 (17:00) ! p5 (19:00) ! p6 (19:30)g. Table 4 shows the result of the candidate routes: T1 - T3 are the original routes and T40 - T60 are three of the reconstructed routes.

Table 3 Raw trajectory dataset.

Tid	(head,tail) subsequence	
T_1	p_1 (10:00)!p_3 (12:00)	p_3 (12:00)!p_5 (15:30)
	p_5 (15:30)!p_8 (17:30)	p_8 (17:30)!p_{10}(19:00)
T_2	p_2 (10:30)!p_3 (12:30)	p_3 (12:30)!p_4 (17:00)
	p_4 (17:00)!p_5 (19:00)	p_5 (19:00)!p_6 (19:30)
T3	p7 (18:30)!p8 (19:30)	p8 (19:30)!p9 (20:00)

Table 4 Subset of Candidate Routes

Tid	POI sequence
1	p1 (10:00)!p3 (12:00)!p5 (15:30)!p8 (17:30)! p10(19:00)
I_2	p2 (10:30)!p3 (12:30)!p4 (17:00)!p5 (19:00)! p6 (19:30)
T3	p7 (18:30)!p8 (19:30)!p9 (20:00)
T	p1 (10:00) p3 (12:30) p4 (17:00) p5 (19:00) p6 (19:30)
T^0	p1 (10:00) p3 (12:00) p5 (19:00) p6 (19:30)
T	1 (10:00) 3 (12:00) 5 (15:30) 8 (19:30) 9 (20:00)

IV. TRAVEL ROUTES EXPLORATION

With the featured trajectory dataset, our final goal is to recommend a set of travel routes that connect to all or partial user-specific keywords. We first explain the matching function to process the user query. Next, we introduce the background of why we apply a skyline query, which is suitable for the travel route recommendation applications, and present the algorithm of the distance-based representative skyline search for the online recommendation system. Furthermore, an approximate algorithm is required to speed up the real-time skyline query. The Travel Route Exploration procedure is presented as Algorithm 2.

Algorithm 2: Travel routes exploration

```
Input: User u, query range Q, a set of keywords K;
Output: Keyword-aware travel routes with diversity in
        goodness domains KRT.
1  Initialize priority queue CR, KRT;
2  Scan the database once to find all candidate routes
   covered by region Q;
   /* Fetch POI scores and check keyword matching */
3  foreach route r found
   do 4 r.kmatch 0;
5     foreach POI p 2 r do
6        r.kmatch r.kmatch + KM(p,k);
7     if r.kmatchthen
8        Push r into CR;
   /* Initialize an arbitrary skyline route, see Section 4.3 */
9  CR.r0  route r with the largest value of an arbitrary
   dimension;
   /* Greedy algorithm for representative skyline, see Algorithm
      3 */
10 KRT  I-greedy(CR);
11 return KRT.
```

4.1 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 arbitrary POI p at query time, we define a keyword matching measure **KM** with the pre-computed information:

$$KM(p; K) = \sum_{w \in K} tf\ idf(w; p)\ (GS(w) + T\ S(w) + AT\ (w)) \quad (5)$$

where tf is the frequency of tag w in a POI and idf is the number of POIs with the tag w. tf idf(w; p) is the product of tf and idf.

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 under score, which means that those routes are not related to the user's preference.

4.2 Representative skyline Travel Routes Search - Given a specific query, we have already retrieved a set of travel routes with multidimensional scores, e.g., attractiveness, time, and geographical social influence scores to fulfill the requirements. To recommend a subset of diverse travel routes, [28] proposed a KST R algorithm applying the skyline search. A skyline search returns the subset of data in a data set which is not dominated by any others. Let a and b be data points, where 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 top-k recommendation system considering a fixed weighting for a set of criteria, skyline query considers all possible weighting criteria that might offer an optimal result, which stands out among others and is of special interest to users. In other words, the results of the skyline travel route are not dominated by any other routes so the user need not specify the weight between every criteria 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 weights 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. However, considering the skyline search may return too many results that are not readable to users, a limitation of a maximum number (an optional k value) of the returned travel routes is required. In the following, we review the existing definition of the distance-based representative skyline in [21], and explain its application over the output of travel routes recommendation.

Definition 2 (Representative skyline travel routes) - Consider the three dimensions that previously mentioned, i.e., attractiveness, time and geographical social influence; trajectory Ti dominates trajectory Tj if and only if the score of Ti in any dimension is not less than the corresponding score of Tj, where i is not equal to j. Given the full skyline S, the representative skyline routes R are the set of routes that has the smallest representation error Er(R; S) among all representative skylines R.

$$Er(R; S) = \max_{p2S\ R} \min_{p\ 2R} k\ p; p^{0}\ kg \qquad (6)$$

4.3 Greedy scoring using multidimensional index

Since computing the optimal representative skyline problem is NP-hard in high dimensional space4, a multidimensional index is helpful to efficiently return the results for real-time applications. Recall that in Section 3.4, the DFS-based approach to generate the candidate routes is to enumerate all subsequences. In the procedure of generation, we can simultaneously build an R-tree index while adding each entry into the dataset Tc (at Line 15 of Algorithm 1). I-greedy [21] is a progressive algorithm that continuously returns 2-approximate guaranteed representative solutions. Instead of retrieving the entire skyline until it is fully computed, I-greedy ables to access only a fraction the sky-line, which saves a considerable cost. The fundamental of I-greedy is the best-first farthest neighbor search. Specifically, given an MBR M in the R-tree, its max representative distance, max-rep-dist(M; R), is a value which upper bounds the representative distance of any potential skyline point p in the subtree of M. Furthermore, to eliminate redundant computations, the greedy algorithm first maintains a conservative skyline based on the intermediate and leaf entries already encountered. Second, it adopts an different access order with fewer empty tests which checks if an arbitrary point is a skyline point.

Conservative skyline - Let O as a mixed set of points and MBRs. A set O 0 is generated with all the points and the side-max corners of the MBRs. The conservative skyline is the skyline of O0. It is proved that any point dominated by the conservative skyline set cannot appear in the real skyline.

Access order - Let L be the set of intermediate and leaf entries that waiting to be processed and E be the entry in L with the largest max-rep-dist. I-greedy checks whether there is any other intermediate of leaf entry in L whose min-corner dominates the min-corner of E, which may result in a tighter conservative skyline.

Algorithm 3 presents the procedure of I-greedy to find out representative skyline results from the candidate routes. The input is the candidate route set containing a skyline route as a point. This point is used as the first representative. Recall that I-greedy does not require a given number of representatives to be returned. Instead, until stopped, it continuously outputs representatives ensuring that their representation error is at most twice larger than the optimal representative skyline of the same size.

Algorithm 3: I-greedy (O)

```
Input: A set O with its arbitrary skyline point O:p0;
Output: Skyline representatives R.
1  Initialize priority queue R;
2  Initialize L to contain the root entries of the R-tree and
   compute Scon of O;
3  while L is not empty do
4      E the entry in L with the largest max-rep-dist;
5      if E is not dominated by any point in Scon then
6          E' = the entry with the minimum L1-distance to the
           origin whose min-corners dominate that of E;
7          if E' exists then
8              access the child node C of E';
9              foreach entry e in C do
10                 if e 6=O:p0 and e is not dominated by any
                   point in Scon then
11                     insert e in L;
12         else
13             if E is a point p then
14                 add p to R;
15             else
16             access the child node C of E;
17             foreach entry e in C do
18                 if e 6=O:p0 and e is not dominated by any
                   point in Scon then
19                     insert e in L;
20 return R.
```

In summary, I-greedy maintains three structures in memory at any mo-ment: (1) the set R of representatives found so far; (2) an access list L that contains all the intermediate and leaf entries that have been encountered but not processed or for the dimensional space that is more than two pruned yet; and (3) a conservative skyline Scon of the set L [R.

Given the set O as the input, I-greedy progressively produces the representatives. At the beginning, L starts with the root entries of the R-tree. Next, I-greedy executes in iterations that identifies the entry E of L with the largest max-rep-dist. Then it checks whether the min-corner of E is dominated by any point in the conservative skyline Scon. If yes, E is pruned, and the current iteration finishes. On the other hand, if E is not pruned, the iteration continues. Following the idea on access order, the entry E^0 with the smallest L1-distance to the origin among all entries in L whose min-corners dominate E needs to be extracted. If E^0 exists, it must be an intermediate entry; otherwise, E would be in the conservative skyline Scon, and would have pruned E already. In this case, the child node of E0 is processed and its entries are inserted into the L that are not dominated by any point in Scon. If E^0 does not exist, I-greedy processes E.

If E is a point, it becomes the next representative skyline point. Otherwise if the points in E are dominated by any point in Scon, we access its child node, and insert its entries in L.

4.3.1 Complexity - Assume that 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 candidate routes in the query range, (ii) calculate feature scores and extract an arbitrary skyline search on all candidate routes, and (iii) derive the representative skyline travel routes. First, the search for (i) takes O(N) and gets even faster since the R-tree based GIS index filters out non-candidate routes efficiently.

Then for each candidate route, step (ii) computes the scores and compares the domination of other routes. The complexity is $O(N^2 l)$. 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 inter-active online system. The process to find out any skyline route with the largest value of an arbitrary dimension takes O(logB N) I/Os where B is the page size. We optimize the implementation by parallelizing the score comparison in step (ii), which involves independent computations of each route. See Section 5.3 for the optimized run time results.

For step (iii), when allowed to run continuously, I-greedy eventually retrieves the whole skyline S with the optimal I/O cost as naive -greedy. Any R-tree-based skyline algorithm must access all nodes whose min -corners are not dominated by any skyline point. Assume that I-greedy is not I/O optimal, and accesses a node M dominated by a skyline point p. This access must happen at either Line 8 or 16 in Algorithm 3. In either case, when M is accessed, p or one of its ancestors must be in L. Otherwise, p already appears in the representative set R, and hence, would have pruned M. As the min-corner of any ancestor of p dominates M, we can eliminate the possibility that M is visited at Line 16, because for this to happen E0 at Line 5 must not exist, i.e., the min-corner of no entry in L can dominate M. On the other hand, if M is visited at Line 8, M must have the lowest L1-distance to the origin, among all entries in L whose min-corners dominate E at Line 3. This is impossible because any E dominated by the min-corner of M is also dominated by p or the min-corner of any of its ancestors, and p or any of its ancestors has a smaller L1-distance to the origin than M.

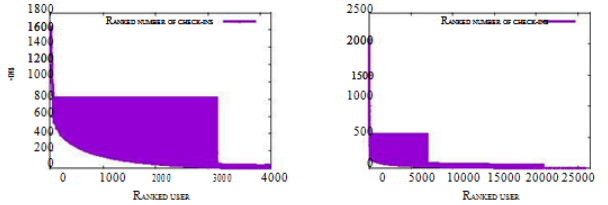
V. EXPERIMENTS

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 4. The FB dataset is collected by Facebook API[5]. 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 [8].

We implemented the system on an x86 64 Linux server with 16 cores and 8 GB memory. All the scores mentioned in Section 3 are computed offline and stored in a PostgreSQL 9.3 database with GIS extension.

Table 5 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



(a) Check-in distribution of the CA (b) Check-in distribution of the FB

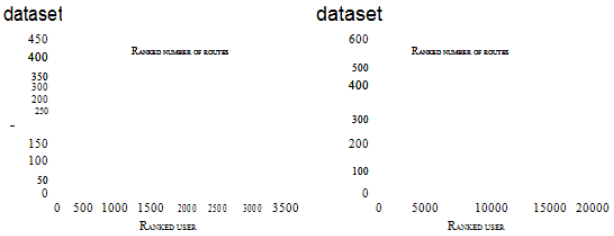


Figure 3. The number of check-ins and the number of routes for all users in the CA and the FB dataset, respectively. The distribution shows a long tail extending in the negative direction.

To gain insights into the datasets, we plotted both the number of check-ins and routes of each user of our datasets. As shown in Figure 3, 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 in the results of inactive users. We therefore chose the top 10% of users, who were ranked by the travel route histories they have, as active users for testing.

5.1 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. For that, the tags are regarded as input keywords. We used Flickr API to collect photos with photo ID, image, location (lat and lon), user ID, photographed time, and textual tags

(only if they existed) as attributes. We collected GPS-tagged photos in the same local area, i.e., the Taipei area6, amounting to 165,057 photos.

We ranked the tags by using the scores in Section 3.1 and measured precision@K. Table 6 shows the precision of deciding the Geo-specific, Temporal, and Attribute keywords.7 We can see that the precision is reasonably high and does not decrease much as K increases. Table 7 shows the results for keyword extraction. Note that the keywords in italics are the Chinese keywords returned, which we translate for presen-tation. In the geo-specific dimension, 10 keywords referring

We set the Taipei area as a rectangle on the globe with left bottom h24:973; 121:423i and right top h25:118; 121:603i.

For attribute extraction, we adopt [15] extracting probable at-tributes of all possible concepts. We can adopt 10 concepts aligned with POI categories, and Table 6 illustrates attributes of the '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 6 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 7 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

In this section, we present the photo and POI datasets, the evaluation measure, and the baselines for evaluation. We used the Flickr dataset amounting to 165,057 photos. We manually matched the photo data with 502 attractions in Taipei obtained from TripAdvisor and, as a result, found 12,463 POI-labeled photos with 64 POIs.

To evaluate the performance of the check-in extraction, we consider a labeled photo as a ground truth check-in hwho: user ID; where: labeled POI; when : photographed timei.

Based on the ground truth, we used the evaluation measures, precision, recall, and F1 score as:

$$\text{precision} = \frac{P_{p2L}\left(I_p^{GT} \cap I_p^{m}\right)}{P_{p2L}\left(I_p^{m}\right)}$$

$$\text{recall} = \frac{P_{p2L}\left(I_p^{GT} \cap I_p^{m}\right)}{P_{p2L}\left(I_p^{GT}\right)}$$

$$F_1 = \frac{2\,\text{precision}\cdot\text{recall}}{\text{precision} + \text{recall}}$$

where IpGT is a set of manually labeled photos on POI p, and Ip^m is a set of photos labeled with p by a check-in extraction method m. We perform a 2-fold validation: Each half of the ground truth is used as training data and test data, respectively.

As candidates for the check-in extraction method m, we present the following two baseline extraction methods, and our three proposed extraction methods.

Base[25]: A baseline method that only considers duplicate/near-duplicate photo clusters with an official POI name.

Base+[25][6]: A baseline method that considers duplicate/near-duplicate photo clusters with multiple POI names extracted by a state-of-the-art name expan-sion method.

SCE: A component, Synonym-based Check-in Extraction, of our proposed method in Section 3.2.1.

CCE: A component, Collective Check-in Extraction, of our proposed method in Section 3.2.2. Note that, to evaluate independently with SCE, CCE uses ne instead of Ne.

SCE + CCE: Our proposed method combining the two components in Section 3.2.

Table 8 Performance of check-in extraction

	Precision	Recall	F1 score
Base	**0.949**	0.437	0.598
Base+	0.935	0.511	0.661
SCE	0.932	0.612	0.739
CCE	0.948	0.467	0.625
SCE + CCE	0.917	**0.696**	**0.791**

Table 8 shows the performance of check-in extraction from Flickr photos. Beyond simple matching with an official POI name, harvesting more check-ins requires a trade-off between precision and recall. The performance of check-in extraction depends on whether this trade-off is well con-trolled. We can see that our proposed method, SCE+CCE, has the best F1 score and a significant recall gain with some loss of precision. The improvement of SCE+CCE is achieved by combining SCE and CCE, which shows the complementary nature of the two components. Base+ (using

synonyms) improves the F1 score and recall compared to Base but not its comparable methods, SCE and SCE+CCE. This fact shows that our scoring for synonym extraction is more effective for POIs.

Because not all web-photos can be used as check-ins, it is an important question how many photos we can use as check-ins. Based on the statistics of datasets and recall performance, we found that our proposed method can use 12;463 69:6% = 5:3% photos as attraction check-ins. 165;057

Considering that five hundred thousand GPS-tagged photos are be-ing uploaded per day by Facebook alone (while geo-tagged photos can be collected from arbitrary sources including Instagram, Twitter, Flickr, and many more), passive check-ins have the potential to complement both the quantity and quality of active check-ins.

As a sensitivity test, Figure 4 shows the performance of check-in extraction (F1 score) when varying threshold and weight parameters and in the two different randomly distributed and same-sized datasets. From the results in Figure 4, we can make the following observations: First, the optimal threshold values are focused on a narrow range, i.e., around 0.8, because the number of POI synonyms is extremely small, e.g., around three in our datasets. Second, around 0.4 to 0.5 is optimal for (linear combination weight for GS or AT). This explains the complementary nature such that our combined approach outperforms using either GS and AT (= 1 or 0). Third, despite the different data distributions, the influence of the parameters used in our approach is very similar in the two heat-maps. This suggests that the supervised learning of and is reliable.

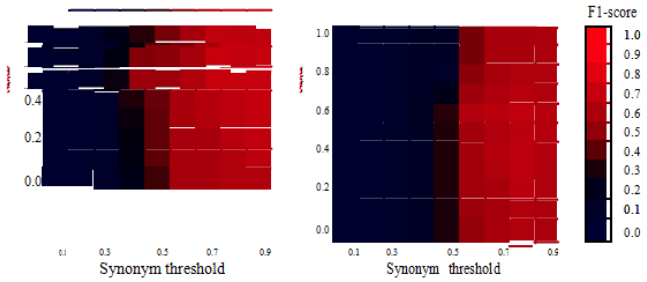


Figure 4 Influence of threshold and weight parameters and The two heat-maps represent F1 scores in different data distributions.

5.2 Evaluation of route prediction accuracy

In this experiment, we compared the following three base-line recommendation models and the original KSTR model with our keyword-aware representative travel route (KRTR) model.

Pattern aware trajectory search (PATS) - Only consider the sum of the POI attractiveness score. Different to the Multinomial model, [26] considers 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 [29]. The route consists of POIs visited by geo-social influential users in the social network. Keyword -aware skyline travel route (KSTR): KSTR [28] outputs full Skyline routes based on both POI and user factors.

Keyword-aware representative travel route (KRTR) - Our KRTR outputs optimal representative Skyline routes.

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 5 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.

5.2.1 Comparison of route prediction accuracy - We measured the difference between the generated routes and each test sequence. Three goodness functions are ap-plied 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 [14]. 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 can both 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 /# of category1 # of category2.

We compared our KRTR model with the other models: KSTR model, pattern aware trajectory search (PATS), time-sensitive (TSR) and geo-social influenced (GSI) routes. Fig-ure 5 shows the performance of each model among the three measures. Overall, we observe that the CA dataset shows better performance than the F B dataset. This might be caused from the fact that the unitary seed users lead to much biased preferences. We can also find that the proposed KRTR model shows near identical results to the KSTR model. Since the output of KRTR is the k-itemset subset of KSTR, we can claim that KRTR is as effective as KSTR without losing the generality, which is the same conclusion as the previous section.

Moreover, it is easy to see that KRTR and KSTR offer the lowest edit distance in both datasets, which represents the highest prediction accuracy. For example, Figure 5(a) depicts that even the worst edit distance results of KRTR is

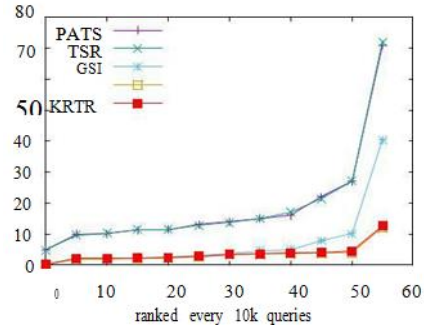
still better than the 90% of the results of the three baseline methods. On the other hand, considering the measure of region cover ratio and category similarity, PATS has better performance in region cover ratio and GSI has better category similarity than ours. The results show that the proposed KRTR is effective and beats other baselines and state-of-the-art methods in terms of **route prediction accuracy.**

5.3 Efficiency

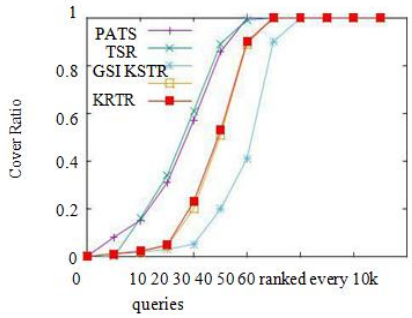
Table 9 shows the online response time of KRTR in the three main sub-procedures: (i) scan the dataset to find the overlap routes and compute the score of candidate routes (O scoring+R scoring), (ii) Initial skyline point search (I skyline), and (iii) Representative skyline search (R skyline). We synthesize 34,928 queries from testing users of the FB dataset and 39,729 queries from the CA dataset. The average response is 1.561708549 seconds. We can find that skyline query (I skyline & R skyline) is the most time-consuming step. In Subsection 5.3.1, we observe the optimal Nfrac for approximate candidate route generation. The total running time under different scales is shown in Subsection 5.3.2.

Table 9 Running time ratio (sub-procedure time cost / total time cost) of each step.

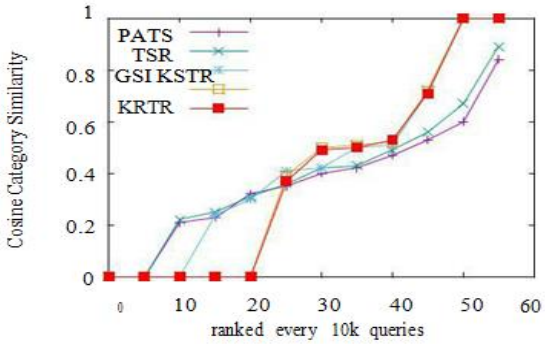
	O scoring	R scoring	I skyline	R skyline
FB	0.160751505	0.040780763	0.099222254	0.265147393
CA	0.155153407	0.038816553	0.163912108	0.211513757



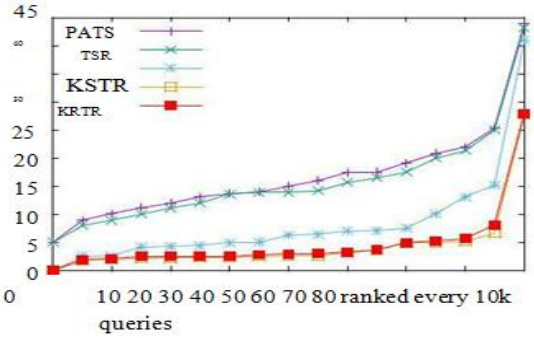
(a) Average edit distance versus to the recommended travel routes of the FB dataset



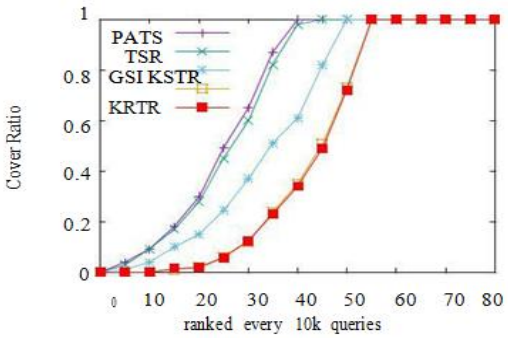
(b) Average region cover ratio versus to the recommended travel routes of the FB dataset



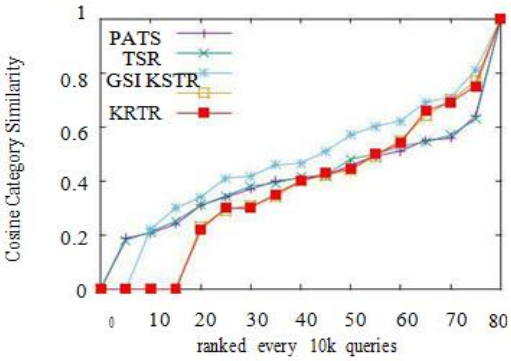
(c) Average category similarity versus to the recommended travel routes of the FB dataset



(d) Average edit distance versus to the recommended travel routes of the CA dataset

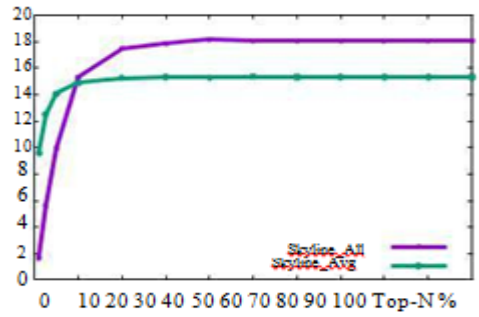


(e) Average region cover ratio versus to the recommended travel routes of the CA dataset

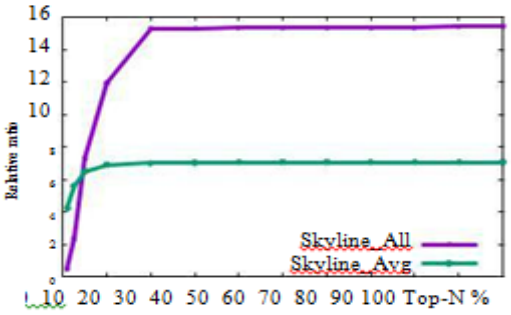


(f) Average category similarity versus to the recommended travel routes of the CA dataset

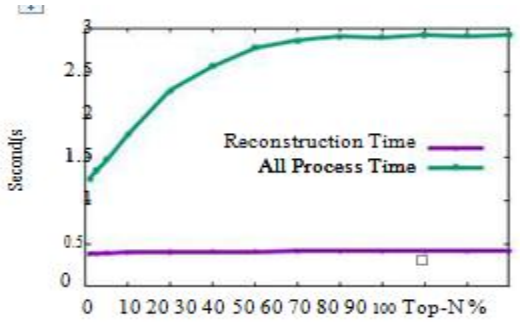
Figure5. Average goodness accuracy of recommended travel route at different query region sizes. The yellow line represents our method and shows that KRTR has good results over the three measurements.



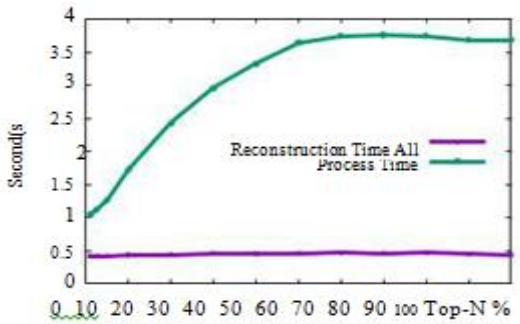
(a) The relative ratio of reconstructed routes in the CA dataset



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



(b) The running time of the reconstruction of the the CA dataset



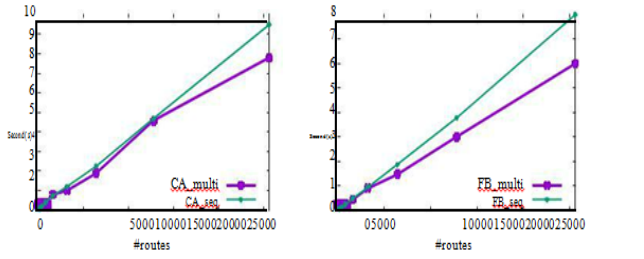
(c) The running time of the reconstruction of the the FB dataset

Figure 6. The effectiveness of the candidate route generation of the CA and FB datasets, respectively under different top-Nf rac% of POI elements. The results converge as R increases.

5.3.1 Tuning Approximation 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 Nfrac for selecting the top-Nfrac% ranked POIs to generate routes that control the best trade-off between effectiveness and running time. Figure 6 shows the average relative ratio of the 1,000 testing routes compared to the value of Nfrac. Note that the brute-force method is N = 100.

As shown in Figures 6(a) and 6(b), we can find that the relative ratio of both datasets converges rapidly as Nfrac increases. Moreover, although the running time of reconstruction is only slightly longer when Nfrac = 100, the running time of the whole procedure is obviously affected because the number of generated routes increases exponentially w.r.t. the size of the POI elements. Moreover, the growth trend of the route number levels off when Nfrac > 50. The reason is that the reconstructed routes start to duplicate when Nfrac is large enough, since the procedure of Candidate Route Generation choose POIs with a high score as elements. Therefore, we choose N frac = 10 in both datasets, which maintains the accuracy and speed.

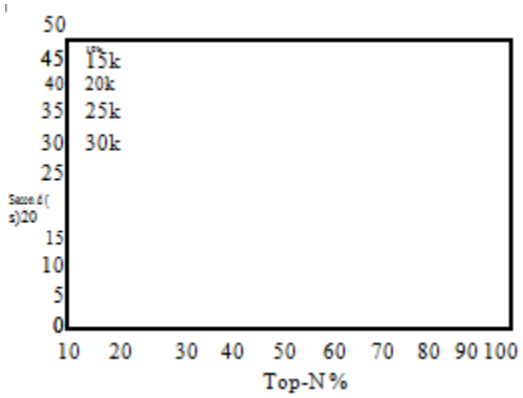
5.3.2 Scalability - The objective of this set of experiments is to study the scalability of the proposed algorithms with variation of the number of computations. We have made use of several methods to optimize the implementation of the online system. Figure 7 shows the total running time and the comparison of the sequential scoring and the multiprocess8 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 Eight-cores multi-processing 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 a multiprocessor reduces 25% of time cost.



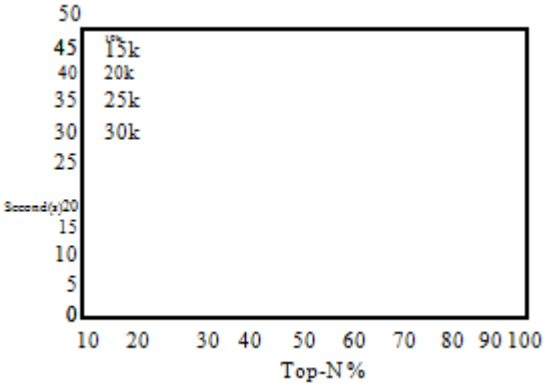
constructed routes in the CA dataset constructed routes in the FB dataset

Figure 7 Runtime versus route number (computation size).

Also, the selection of Nfrac is fixed to 10 within a larger route processing number. As shown in Figure 8, the average results of 100 queries within 10k to 30k candidate routes. The curves present similar trends to Figures 6(c) and 6(d).



(a) The process time under different query numbers of the CA dataset



(b) The process time under different query numbers of the FB dataset

Figure 8 The total process time of the candidate route generation under different top-Nf rac% of POI elements.

Trip Planning - Trip planning has been intensively studied recently. The problem is to develop a collaborative recommendation model to recommend routes for a given user at a query region. Some studies have modeled the goodness of existing trip routes by self-defined traveling factors [26] [35][40]. On the other hand, [17] [13] [12] [11] constructed personalized routes according to user queries. The traveling factors can be summarized into "Where, When, Who" issues. For example, [12] and [11] developed a system to construct time-sensitive routes, which considered location popularity, visiting order, proper visiting time, and proper transit time to model the goodness of a route. [17] developed the Photo2Trip system, which integrates a series of traveling factors including time duration, season, user preference, destination type, and popularity to recommend trip itineraries. [13] ranked the constructed routes by the location attractiveness, proper visiting time and the distance to query locations.

Location Recommendation and Prediction- In addition, a number of research projects focused on recommendation and prediction of single location. The task of location recommendation is to recommend new locations that the user has never visited before.

VI. CONCLUSION

In this paper, we study the travel route recommendation problem. We have developed a KRTR framework to suggest travel routes with a specific range and a set of user preference keywords. These travel routes are related to all or partial user preference keywords, and are recommended based on (i) the attractiveness of the POIs it passes, (ii) visiting the POIs at their corresponding proper arrival times, and (iii) the routes generated by influential users. We propose a novel keyword extraction module to identify the semantic meaning and match the measurement of routes, and have designed a route reconstruction algorithm to aggregate route segments into travel routes in accordance with query range and time period. We leverage score functions for the three aforementioned features and adapt the representative Skyline search instead of the traditional top-k recommendation system. The experiment results demonstrate that KRTR is able to retrieve travel routes that are interesting for users, and outperforms the baseline algorithms in terms of effectiveness and efficiency. Due to the real-time requirements for online systems, we aim to reduce the computation cost by recording repeated queries and to learn the approximate parameters automatically in the future.

VII. REFERENCES

[1]. Y. Arase, X. Xie, T. Hara, and S. Nishio. Mining people's trips from large scale geo-tagged photos. In Proceedings of the 18th ACM international conference on Multimedia, pages 133–142.ACM, 2010.

[2]. X. Cao, L. Chen, G. Cong, and X. Xiao. Keyword-aware optimal route search. Proceedings of the VLDB Endowment, 5(11):1136– 1147, 2012.

[3]. X. Cao, G. Cong, and C. S. Jensen. Mining significant semantic locations from GPS data. Proceedings of the VLDB Endowment, 3(1-2):1009–1020, 2010.

[4]. D. Chen, C. S. Ong, and L. Xie. Learning points and routes to rec-ommend trajectories. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, pages 2227– 2232, 2016.

[5]. Z. Chen, H. T. Shen, X. Zhou, Y. Zheng, and X. Xie. Searching trajectories by locations: an efficiency study. In Proceedings of the 2010 ACM SIGMOD International Conference on Management of data, pages 255–266, 2010.

[6]. T. Cheng, H. W. Lauw, and S. Paparizos. Entity synonyms for structured web search. IEEE transactions on knowledge and data engineering, 24(10):1862–1875, 2012.

[7]. M.-F. Chiang, Y.-H. Lin, W.-C. Peng, and P. S. Yu. Inferring distant-time location in low-sampling-rate trajectories. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1454–1457. ACM, 2013.

[8]. H. Gao, J. Tang, and H. Liu. Exploring social-historical ties on location-based social networks. In ICWSM, 2012.

[9]. Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 899–908, 2010.

[10]. F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pat-tern mining. In Proceedings of the 13th ACM

SIGKDD international conference on Knowledge discovery and data mining, pages 330–339, 2007.

[11]. H.-P. Hsieh and C.-T. Li. Mining and planning time-aware routes from check-in data. In Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, pages 481–490, 2014.

Mr. Vishal Rokade

Mr. Gaurav Kulkarni

Ms. Shatavari Raut

Mr. Anurag Katkam

Prof. Navnath Bagal