Learning Points and Routes to Recommend Trajectories

Dawei Chen*†, Cheng Soon Ong†*, Lexing Xie*†
*The Australian National University, †Data 61, CSIRO, Australia
{u5708856, chengsoon.ong, lexing.xie}@anu.edu.au

Abstract

The problem of recommending tours to travellers is an important and broadly studied area. Suggested solutions include various approaches of points-of-interest (POI) recommendation and route planning. We consider the task of recommending a sequence of POIs, that simultaneously uses information about POIs and routes. Our approach unifies the treatment of various sources of information by representing them as features in machine learning algorithms, enabling us to learn from past behaviour. Information about POIs are used to learn a POI ranking model that accounts for the start and end points of tours. Data about previous trajectories are used for learning transition patterns between POIs that enable us to recommend probable routes. In addition, a probabilistic model is proposed to combine the results of POI ranking and the POI to POI transitions. We propose a new F_1 score on pairs of POIs that capture the order of visits. Empirical results show that our approach improves on recent methods, and demonstrate that combining points and routes enables better trajectory recommendations.

Keywords

Trajectory recommendation; learning to rank; planning

1 Introduction

This paper proposes a novel solution to recommend travel routes in cities. A large amount of location traces are becoming available from ubiquitous location tracking devices. For example, FourSquare has 50 million monthly users who have made 8 billion check-ins [7], and Flickr hosts over 2 billion geo-tagged public photos [6]. This growing trend in rich geolocation data provide new opportunities for better travel planning traditionally done with written travel guides. Good solutions to these problems will in turn lead to better urban experiences for residents and visitors alike, and foster sharing of even more location-based behavioural data.

There are several settings of recommendation problems for locations and routes, as illustrated in Figure 1. We summarise recent work most related to formulating and solving learning problems on assembling routes from POIs, and refer the reader to a number of recent surveys [1, 27, 28] for general overviews of the area. The first setting can be called POI recommendation (Figure 1(a)). Each location (A to E) is scored with geographic and behavioural information such as category, reviews, popularity, spatial information such as distance, and temporal information such as travel time uncertainty, time of the day or day of the week. A popular approach is to recommend POIs with a collaborative filtering model on user-location affinity [21], with additional ways to incorporate spatial [13, 16], temporal [24, 11, 8], or spatial-temporal [25] information.

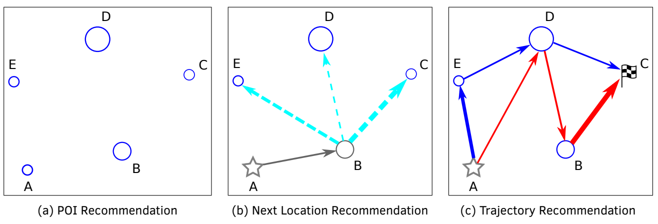

D
E
C
A
B
(a) POI Recommendation
(b) Next Location Recommendation
(c) Trajectory Recommendation

Figure 1: Three settings of trajectory recommendation problems. Node size: POI score; edge width: transition score between pairs of POIs; grey: observed; star: starting location; flag: ending location. See Section 1 for details.

Figure 1(b) illustrates the second setting: next location recommendation. Here the input is a partial trajectory (e.g. started at point A and currently at point B), the task of the algorithm is to score the next candidate location (e.g, C, D and E) based on the perceived POI score and transition compatibility with input $A \to B$. It is a variant of POI recommendation except both the user and locations travelled to date are given. The solutions to this problem include incorporating Markov chains into collaborative filtering [20, 4, 26], quantifying tourist traffic flow between points-of-interest [29], formulating a binary decision or ranking problem [2], and predict the next location with sequence models such as recurrent neural networks [15].

This paper considers the final setting: trajectory recommendation (Figure 1(c)). Here the input are some factors about the desired route, e.g. starting point A and end point C, along with auxiliary information such as the desired length of trip. The algorithm needs to take into account location desirability (as indicated by node size) and transition compatibility (as indicated by edge width), and compare route hypotheses such as A-D-B-C and A-E-D-C. Existing work in this area either uses heuristic combination of locations and routes [18, 14, 17], or formulates an optimisation problem that is not informed or evaluated by behaviour history [9, 3]. We note, however, that two desired qualities are still missing from the current solutions to trajectory recommendation. The first is a principled method to jointly learn POI ranking (a prediction problem) and optimise for route creation (a planning problem). The second is a unified way to incorporate various features such as location, time, distance, user profile and social interactions, as they tend to get specialised and separate treatments. This work aims to address both challenges. We propose a novel way to learn point preferences and routes jointly. In Section 2, we describe the features that are used to ranking points, and POI to POI transitions that are factorised along different types of location properties. Section 3 details a number of our proposed approaches to recommend trajectories. We evaluate the proposed algorithms on trajectories from five different cities in Section 4. The main contributions of this work are:

- We propose a novel algorithm to jointly optimise point preferences and routes. We find that learning-based approaches generally outperform heuristic route recommendation [14]. Incorporating transitions to POI ranking results in a better sequence of POIs, and avoiding sub-tours further improves performance of classical Markov chain methods.
- Our approach is feature-driven and learns from past behaviour without having to design specialised treatment for spatial, temporal or social information. It incorporates information about location, POI categories and behaviour history, and can use additional time, user, or social information if available.
- We show good performance compared to recent results [14], and also quantify the contributions from different components, such as ranking points, scoring transitions, and routing.
- We propose a new metric to evaluate trajectories, pairs-F_1, to capture the order in which POIs are visited. Pairs-F_1 lies between 0 and 1, and achieves 1 if and only if the recommended trajectory is exactly the same as the ground truth.

Supplemental material, benchmark data and results are available online at https://bitbucket.org/d-chen/tour-cikm16 .

2 POI, Query and Transition

The goal of tour recommendation is to suggest a sequence of POIs, $(p_1, \dots, p_L)$, of length L such that the user's utility is maximised. The user provides the desired start $(p_1 = p_s)$ and end point $(p_L = p_e)$, as well as the number L of POIs desired, from which we propose a trajectory through the city. The training data consists of a set of tours of varying length in a particular city. We consider only POIs that have been visited by at least one user in the past, and construct a graph with POIs as nodes and directed edges representing the observed transitions between pairs of POIs in tours.

We extract the category, popularity (number of distinct visitors) [5], total number of visits and average visit duration for each POI. POIs are grouped into 5 clusters using K-means according to their geographical locations to reflect their neighbourhood. Furthermore, since we are constrained by the fact that trajectories have to be of length L and start and end at certain points, we hope to improve the recommendation by using this information. In other words, we use the *query* $q = (p_s, p_e, L)$ to construct new features by contrasting candidate POIs with p_s and p_e. For each of the POI features (category, neighbourhood, popularity, total visits and average duration), we construct two new features by taking the difference of the feature in POI p with p_s and p_e respectively. For the category (and neighbourhood), we set the feature to 1 when their categories (and cluster identities) are the same and -1 otherwise. For popularity, total visits and average duration, we take the real valued difference. Lastly, we compute the distance from POI p to p_s (and p_e) using the Haversine formula [22], and also include the required length L.

In addition to information about each individual POI, a tour recommendation system would benefit from capturing the likelihood of going from one POI to another different POI. One option would be to directly model the probability of going from any POI to any other POI, but this has several weaknesses: Such a model would be unable to handle a new POI (one that has not yet been visited), or pairs of existing POIs that do not have an observed transition. Furthermore, even if we restrict ourselves to known POIs and transitions, there may be locations which are rarely visited, leading to significant challenges in estimating the probabilities from empirical data.

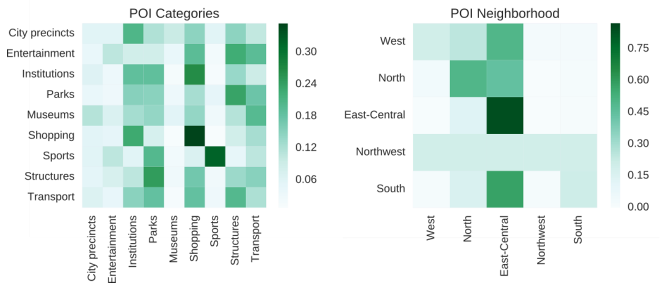


Figure 2: Transition matrices for two POI features from Melbourne: POI category and neighbourhood.

We model POI transitions using a Markov chain with discrete states by factorising the transition probability (p_i to p_j) as a product of transition probabilities between pairs of individual POI features, assuming independence between these feature-wise transitions. The popularity, total visits and average duration are discretised by binning them uniformly into 5 intervals on the log scale. These feature-to-feature transitions are estimated from data using maximum likelihood principle. The POI-POI transition probabilities can be efficiently computed by taking the Kronecker product of transition matrices for the individual features, and then updating it based on three additional constraints as well as appropriate normalisation. First we disallow self-loops by setting the probability of (p_i to p_i) to zero. Secondly, when multiple POIs have identical (discretised) features, we distribute the probability uniformly among POIs in the group. Third, we remove feature combinations that has no POI in dataset. Figure 2 visualises the transition matrices for two POI features, category and neighbourhood, in Melbourne.

3 Tour Recommendation

In this section, we first describe the recommendation of points and routes, then we discuss how to combine them, and finally we propose a method to avoid sub-tours.

3.1 POI Ranking and Route Planning

A naive approach would be to recommend trajectories based on the popularity of POIs only, that is we always suggest the top-k most popular POIs for all visitors given the start and end location. We call this baseline approach PoiPopularity, and its only adaptation to a particular query is to adjust k to match the desired length.

On the other hand, we can leverage the whole set of POI features described in Section 2 to learn a ranking of POIs using rankSVM, with linear kernel and L2 loss [12],

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_{p_i, p_j \in \mathcal{P},\ q \in \mathcal{Q}} \max\left(0,\ 1 - \mathbf{w}^T(\phi_{i,q} - \phi_{j,q})\right)^2,$$

where $\mathbf{w}$ is the parameter vector, $C > 0$ is a regularisation constant, $\mathcal{P}$ is the set of POIs to rank, $\mathcal{Q}$ denotes the queries corresponding to trajectories in training set, and $\phi_{i,q}$ is the feature vector for POI p_i with respect to query q. The ranking score of p_i given query q is computed as $R_{i,q} = \mathbf{w}^T\phi_{i,q}$.

For training the rankSVM, the labels are generated using the number of occurrences of POI p in trajectories grouped by query (p_s, p_e, L), without counting the occurrence of p when it is the origin or destination of a trajectory. Our algorithm, PoiRank, recommends a trajectory for a particular query by first ranking POIs then takes the top ranked $L-2$ POIs and connects them according to the ranks.

In addition to recommend trajectory by ranking POIs, we can leverage the POI-POI transition probabilities and recommend a trajectory (with respect to a query) by maximising the transition likelihood. The maximum likelihood of the Markov chain of transitions is found using a variant of the Viterbi algorithm (with uniform emission probabilities). We call this approach that only uses the transition probabilities between POIs as Markov.

3.2 Combine Ranking and Transition

We would like to leverage both point ranking and transitions, i.e., recommending a trajectory that maximises the points ranking of its POIs as well as its transition likelihood at the same time. To begin with, we transform the ranking scores $R_{j,q}$ of POI p_j with respect to query q to a probability distribution using the softmax function,

$$P_R(p_j|q) = \frac{\exp(R_{j,q})}{\sum_i \exp(R_{i,q})}, \qquad (1)$$

One option to find a trajectory that simultaneously maximises the ranking probabilities of its POIs and its transition likelihood is to optimise the following objective:

$$\operatorname*{argmax}_{\mathcal{T} \in \mathcal{P}^L} \; \alpha \sum_{k=2}^{L} \log P_R(p_k|q) + (1-\alpha) \sum_{k=1}^{L-1} \log P(p_{k+1}|p_k),$$

such that $p_1 = p_s,\ p_L = p_e$ and $p_k \in \mathcal{P},\ 1 \le k \le L$. The first term captures the POI ranking, and the second one incorporates the transition probabilities. $\mathcal{T} = (p_1, \dots, p_L)$ is any possible trajectory, $\alpha \in [0,1]$ is a parameter to trade-off the importance between point ranking and transition, and can be tuned using cross validation in practice. Let $S(p; p', q)$ be a convex combination of point ranking and transition,

$$S(p; p', q) = \alpha \log P_R(p|q) + (1-\alpha) \log P(p|p'), \qquad (2)$$

then the best path (or walk) can be found using the Viterbi algorithm. We call this approach that uses both the point ranking and transitions Rank+Markov, with pseudo code shown in Algorithm 1, where A is the score matrix, and entry $A[l,p]$ stores the maximum value associated with the (partial) trajectory that starts at p_s and ends at p with l POI visits; B is the backtracking-point matrix, and entry $B[l,p]$ stores the predecessor of p in that (partial) trajectory. The maximum objective value is $A[L, p_e]$, and the corresponding trajectory can be found by tracing back from $B[L, p_e]$.

Algorithm 1 Rank+Markov: recommend trajectory with POI ranking and transition

```
 1: Input: $\mathcal{P}, p_s, p_e, L$
 2: Output: Trajectory $\mathcal{T} = (p_s, \cdots, p_e)$ with $L$ POIs
 3: Initialise score matrix $A$ and backtracking pointers $B$
 4: for $p \in \mathcal{P}$ do
 5:     $A[2,p] = S(p; p_s, q)$
 6:     $B[2,p] = p_s$
 7: end for
 8: for $l = 2$ to $L-1$ do
 9:     for $p \in \mathcal{P}$ do
10:         $A[l+1,p] = \max_{p' \in \mathcal{P}} \{A[l,p'] + S(p; p', q)\}$
11:         $B[l+1,p] = \operatorname{argmax}_{p' \in \mathcal{P}} \{A[l,p'] + S(p; p', q)\}$
12:     end for
13: end for
14: $\mathcal{T} = \{p_e\}$, $l = L$, $p = p_e$
15: repeat
16:     Prepend $B[l,p]$ to $\mathcal{T}$
17:     $l = l-1$, $p = B[l,p]$
18: until $l < 2$
19: return $\mathcal{T}$
```

3.3 Avoiding sub-tours

Trajectories recommended by Markov (Section 3.1) and Rank+Markov (Section 3.2) are found using the maximum likelihood approach, and may contain multiple visits to the same POI. This is because the best solution from Viterbi decoding may have circular sub-tours (where a POI already visited earlier in the tour is visited again). We propose a method for eliminating sub-tours by finding the best path using an integer linear program (ILP), with sub-tour elimination constraints adapted from the Travelling Salesman Problem [19]. In particular, given a set of POIs $\mathcal{P}$, the POI-POI transition matrix and a query $q = (p_s, p_e, L)$, we recommend a trajectory by solving the following ILP:

$$\max_{x,u} \sum_{i=1}^{N-1} \sum_{j=2}^{N} x_{ij} \; \log P(p_j|p_i)$$

$$s.t. \quad x_{ij} \in \{0,1\},\ x_{ii} = 0,\ u_i \in \mathbf{Z},\ \forall i,j = 1, \cdots, N \qquad (3)$$

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1,\ \sum_{i=2}^{N} x_{i1} = \sum_{j=1}^{N-1} x_{Nj} = 0 \qquad (4)$$

$$\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \le 1,\ \forall k = 2, \cdots, N-1 \qquad (5)$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} x_{ij} = L-1, \qquad (6)$$

$$u_i - u_j + 1 \le (N-1)(1-x_{ij}), \forall i,j = 2, \cdots, N \qquad (7)$$

where $N = |\mathcal{P}|$ is the number of available POIs and x_{ij} is a binary decision variable that determines whether the transition from p_i to p_j is in the resulting trajectory. For brevity, we arrange the POIs such that $p_1 = p_s$ and $p_N = p_e$. Firstly, the desired trajectory should start from p_s and end at p_e (Constraint 4). In addition, any POI could be visited at most once (Constraint 5). Moreover, only $L-1$ transitions between POIs are permitted (Constraint 6), i.e., the number of POI visits should be exactly L (including p_s and p_e). The last constraint, where u_i is an auxiliary variable, enforces that only a single sequence of POIs without sub-tours is permitted in the trajectory. We solve this ILP using the Gurobi optimisation package [10], and the resulting trajectory is constructed by tracing the non-zeros in x. We call our method that uses the POI-POI transition matrix to recommend paths without circular sub-tours MarkovPath.

Sub-tours in trajectories recommended by Rank+Markov can be eliminated in a similar manner, we solve an ILP by optimising the following objective with the same constraints described above,

$$\max_{x,u} \sum_{i=1}^{N-1} \sum_{j=2}^{N} x_{ij} \; S(p_j; p_i, q), \qquad (8)$$

where $S(p_j; p_i, q)$ incorporates both point ranking and tran-

Table 1: **Statistics of trajectory dataset**

Dataset	#Photos	#Visits	#Traj.	#Users
Edinburgh	82,060	33,944	5,028	1,454
Glasgow	29,019	11,434	2,227	601
Melbourne	94,142	23,995	5,106	1,000
Osaka	392,420	7,747	1,115	450
Toronto	157,505	39,419	6,057	1,395

Table 2: Summary of information captured by different trajectory recommendation algorithms

	Query	POI	Trans.	No sub-tours
RANDOM	×	×	×	×
PERSTOUR[14]	×	✓	×	✓
PERSTOUR-L	×	✓	×	✓
POIPOPULARITY	×	✓	×	×
POIRANK	✓	✓	×	×
MARKOV	×	✓	✓	×
MARKOVPATH	×	✓	✓	✓
RANK+MARKOV	✓	✓	✓	×
RANK+MARKOVPATH	✓	✓	✓	✓

sition, as defined in Equation (2). This algorithm is called RANK+MARKOVPATH in the experiments.

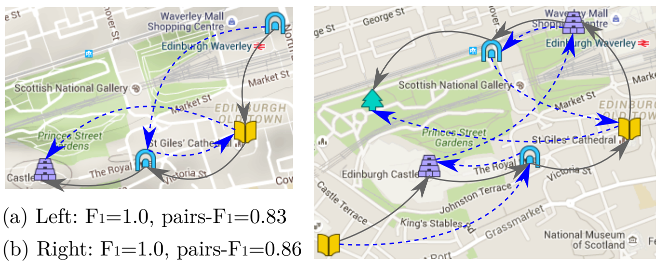
4 Experiment on Flickr Photos

We evaluate the algorithms above on datasets with trajectories extracted from Flickr photos [23] in five cities, namely, Edinburgh, Glasgow, Melbourne, Osaka and Toronto, with statistics shown in Table 1. The Melbourne dataset is built using approaches proposed in earlier work [5, 14], and the other four datasets are provided by Lim et al. [14].

We use leave-one-out cross validation to evaluate different trajectory recommendation algorithms, i.e., when testing on a trajectory, all other trajectories are used for training. We compare with a number of baseline approaches such as RANDOM, which naively chooses POIs uniformly at random (without replacement) from the set $\mathcal{P} \setminus \{p_s, p_e\}$ to form a trajectory, and POIPOPULARITY (Section 3.1), which recommends trajectories based on the popularity of POIs only. Among the related approaches from recent literature, PERSTOUR [14] explores POI features as well as the sub-tour elimination constraints (Section 3.3), with an additional time budget, and its variant PERSTOUR-L, which replaces the time budget with a constraint of trajectory length. Variants of point-ranking and route-planning approaches including POIRANK and MARKOV (Section 3.1), which utilises either POI features or POI-POI transitions, and RANK+MARKOV (Section 3.2) that captures both types of information. Variants that employ additional sub-tour elimination constraints (MARKOVPATH and RANK+MARKOVPATH, Section 3.3) are also included. A summary of the various trajectory recommendation approaches can be found in Table 2.

4.1 Performance metrics

A commonly used metric for evaluating POI and trajectory recommendation is the F_1 score on points, which is the harmonic mean of precision and recall of POIs in trajectory [14]. While being good at measuring whether POIs are correctly recommended, F_1 score on points ignores the visiting order



(a) Left: F_1=1.0, pairs-F_1=0.83
(b) Right: F_1=1.0, pairs-F_1=0.86

Figure 3: Examples for F_1 vs pairs-F_1 as evaluation metric. Solid grey: ground truth; dashed blue: recommended trajectories. See Section 4.1 for details.

between POIs. We propose a new metric pairs-F_1 that considers both POI identity and visiting order, by measuring the F_1 score of every pair of POIs, whether they are adjacent or not in trajectory,

$$\text{pairs-F}_1 = \frac{2 P_{\text{PAIR}} R_{\text{PAIR}}}{P_{\text{PAIR}} + R_{\text{PAIR}}},$$

where P_{PAIR} and R_{PAIR} are the precision and recall of ordered POI pairs respectively. Pairs-F_1 takes values between 0 and 1 (higher is better). A perfect pairs-F_1 is achieved *if and only if* both the POIs and their visiting order in the recommended trajectory are exactly the same as those in the ground truth. On the other hand, pairs-$F_1 = 0$ means none of the recommended POI pairs was actually visited (in the designated order) in the real trajectory. An illustration is shown in Figure 3, the solid grey lines represent the ground truth transitions that actually visited by travellers, and the dashed blue lines are the recommended trajectory by one of the approaches described in Section 3. Both examples have a perfect F_1 score, but not a perfect pairs-F_1 score due to the difference in POI sequencing.

4.2 Results

The performance of various trajectory recommendation approaches are summarised in Table 3 and Table 4, in terms of F_1 and pairs-F_1 scores respectively. It is apparent that algorithms captured information about the problem (Table 2) outperform the RANDOM baseline in terms of both metrics on all five datasets.

Algorithms based on POI ranking yield strong performance, in terms of both metrics, by exploring POI and query specific features. POIRANK improves notably upon POIPOPULARITY and PERSTOUR by leveraging more features. In contrast, MARKOV which leverages only POI transitions does not perform as well. Algorithms with ranking information (RANK+MARKOV and RANK+MARKOVPATH) always outperform their respective variants with transition information alone (MARKOV and MARKOVPATH).

We can see from Table 3 that, in terms of F_1, MARKOVPATH and RANK+MARKOVPATH outperform their corresponding variants MARKOV and RANK+MARKOV without the path constraints, which demonstrates that eliminating sub-tours improves point recommendation. This is not unexpected, as sub-tours worsen the proportion of correctly recommended POIs since a length constraint is used. In contrast, most Markov chain entries have better performance in terms of pairs-F_1 (Table 4), which indicates Markov chain approaches generally respect the transition patterns between POIs.

PERSTOUR [14] always performs better than its variant PERSTOUR-L, in terms of both metrics, especially on Glas-

Table 3: Performance comparison on five datasets in terms of F_1 score. The best method for each dataset (i.e., a column) is shown in bold, the second best is shown in *italic*.

	Edinburgh	Glasgow	Melbourne	Osaka	Toronto
RANDOM	0.570 ± 0.139	0.632 ± 0.123	0.558 ± 0.149	0.621 ± 0.115	0.621 ± 0.129
PERSTOUR[14]	0.656 ± 0.223	**0.801 ± 0.213**	0.483 ± 0.208	0.686 ± 0.231	0.720 ± 0.215
PERSTOUR-L	0.651 ± 0.143	0.660 ± 0.102	0.576 ± 0.141	0.686 ± 0.137	0.643 ± 0.113
POIPOPULARITY	**0.701 ± 0.160**	0.745 ± 0.166	0.620 ± 0.136	0.663 ± 0.125	0.678 ± 0.121
POIRANK	*0.700 ± 0.155*	*0.768 ± 0.171*	*0.637 ± 0.142*	**0.745 ± 0.173**	**0.754 ± 0.170**
MARKOV	0.645 ± 0.169	0.725 ± 0.167	0.577 ± 0.168	0.697 ± 0.150	0.669 ± 0.151
MARKOVPATH	0.678 ± 0.149	0.732 ± 0.168	0.595 ± 0.148	0.706 ± 0.150	0.688 ± 0.138
RANK+MARKOV	0.659 ± 0.174	0.754 ± 0.173	0.613 ± 0.166	0.715 ± 0.164	0.723 ± 0.185
RANK+MARKOVPATH	0.697 ± 0.152	0.762 ± 0.167	**0.639 ± 0.146**	*0.732 ± 0.162*	*0.751 ± 0.170*

Table 4: Performance comparison on five datasets in terms of pairs-F_1 score. The best method for each dataset (i.e., a column) is shown in bold, the second best is shown in *italic*.

	Edinburgh	Glasgow	Melbourne	Osaka	Toronto
RANDOM	0.261 ± 0.155	0.320 ± 0.168	0.248 ± 0.147	0.304 ± 0.142	0.310 ± 0.167
PERSTOUR[14]	0.417 ± 0.343	**0.643 ± 0.366**	0.216 ± 0.265	0.468 ± 0.376	0.504 ± 0.354
PERSTOUR-L	0.359 ± 0.207	0.352 ± 0.162	0.266 ± 0.140	0.406 ± 0.238	0.333 ± 0.163
POIPOPULARITY	*0.436 ± 0.259*	0.507 ± 0.298	0.316 ± 0.178	0.365 ± 0.190	0.384 ± 0.201
POIRANK	0.432 ± 0.251	*0.548 ± 0.311*	0.339 ± 0.203	**0.511 ± 0.309**	**0.518 ± 0.296**
MARKOV	0.417 ± 0.248	0.495 ± 0.296	0.288 ± 0.195	0.445 ± 0.266	0.407 ± 0.241
MARKOVPATH	0.400 ± 0.235	0.485 ± 0.293	0.294 ± 0.187	0.442 ± 0.260	0.405 ± 0.231
RANK+MARKOV	**0.444 ± 0.263**	0.545 ± 0.306	**0.351 ± 0.220**	0.486 ± 0.288	0.512 ± 0.303
RANK+MARKOVPATH	0.428 ± 0.245	0.533 ± 0.303	*0.344 ± 0.206*	*0.489 ± 0.287*	*0.514 ± 0.297*

gow and Toronto datasets. This indicates the time budget constraint is more helpful than length constraint for recommending trajectories. Surprisingly, we observed that PERSTOUR is outperformed by RANDOM baseline on Melbourne dataset. It turns out that on this dataset, many of the ILP problems which PERSTOUR needs to solve to get the recommendations are difficult ILP instances. In the leave-one-out evaluation, although we utilised a large scale computing cluster with modern hardware, 12% of evaluations failed as the ILP solver was unable to find a feasible solution after 2 hours. Furthermore, a lot of recommendations were suboptimal solutions of the corresponding ILPs due to the time limit. These factors lead to the inconsistent performance of PERSTOUR on Melbourne dataset.

4.3 An Illustrative Example

Figure 4 illustrates an example from Edinburgh. The ground truth is a trajectory of length 4 that starts at a POI of category *Structures*, visits two intermediate POIs of category *Structures* and *Cultural* and terminates at a POI of category *Structures*. The trajectory recommended by PERSTOUR is a tour with 11 POIs, as shown in Figure 4(a), with none of the desired intermediate POIs visited. POIRANK (Figure 4(b)) recommended a tour with correct POIs, but with completely different routes. On the other hand, MARKOV (Figure 4(c)) missed one POI but one of the intermediate routes is consistent with the ground truth. The best recommendation, as shown in Figure 4(d), with exactly the same points and routes as the ground truth, which in this case is achieved by RANK+MARKOVPATH.

5 Discussion and Conclusion

In this paper, we propose an approach to recommend trajectories by jointly optimising point preferences and routes. This is in contrast to related work which looks at only POI or next location recommendation. Point preferences are learned by ranking according to POI and query features, and factorised transition probabilities between POIs are learned from previous trajectories extracted from social media. We investigate the maximum likelihood sequence approach (which may recommend sub-tours) and propose an improved sequence recommendation method. Our feature driven approach naturally allows learning the combination of POI ranks and routes.

We argue that one should measure performance with respect to the visiting order of POIs, and suggest a new pairs-F_1 metric. We empirically evaluate our tour recommendation approaches on five datasets extracted from Flickr photos, and demonstrate that our method improves on prior work, in terms of both the traditional F_1 metric and our proposed performance measure. Our promising results from learning points and routes for trajectory recommendation suggests that research in this domain should consider both information sources simultaneously.

Acknowledgements

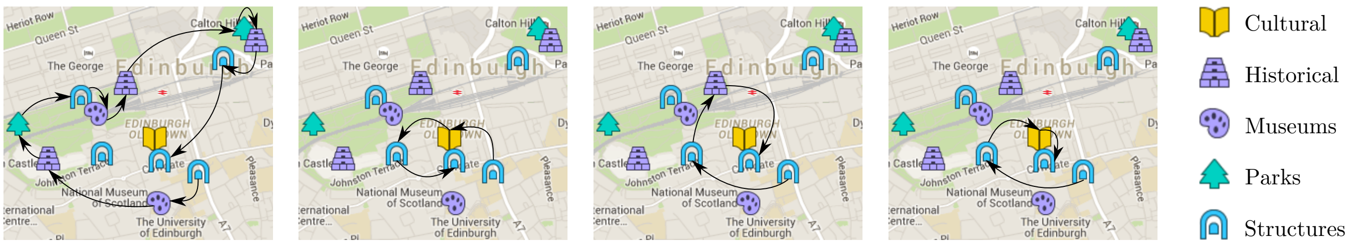
We thank Kwan Hui Lim for kindly providing his R code to reproduce his experiments. This work is supported in part by the Australian Research Council via the Discovery Project program DP140102185.

References

[1] J. Bao, Y. Zheng, D. Wilkie, and M. Mokbel. Recommendations in location-based social networks: a survey. *GeoInformatica*, 19(3):525–565, 2015.

[2] R. Baraglia, C. I. Muntean, F. M. Nardini, and F. Silvestri. LearNext: learning to predict tourists movements. CIKM '13, pages 751–756. ACM, 2013.

[3] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu. TRIPPLANNER: Personalized trip planning lever-



(a) PersTour (b) PoiRank (c) Markov (d) Best

Figure 4: Different recommendations from algorithm variants. See the main text in Section 4.3 for description.

aging heterogeneous crowdsourced digital footprints. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1259–1273, 2015.

[4] C. Cheng, H. Yang, M. R. Lyu, and I. King. Where you like to go next: Successive point-of-interest recommendation. IJCAI '13, pages 2605–2611. AAAI Press, 2013.

[5] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. HT '10, pages 35–44. ACM, 2010.

[6] Flickr. Flickr photos on the map. https://www.flickr.com/map , retrieved May 2016.

[7] Foursquare. FourSquare: about us. https://foursquare.com/about , retrieved May 2016.

[8] H. Gao, J. Tang, X. Hu, and H. Liu. Exploring temporal effects for location recommendation on location-based social networks. RecSys '13, pages 93–100. ACM, 2013.

[9] A. Gionis, T. Lappas, K. Pelechrinis, and E. Terzi. Customized tour recommendations in urban areas. WSDM '14, pages 313–322. ACM, 2014.

[10] Gurobi. Gurobi Optimization. http://www.gurobi.com , retrieved May 2016.

[11] H.-P. Hsieh and C.-T. Li. Mining and planning time-aware routes from check-in data. CIKM '14, pages 481–490. ACM, 2014.

[12] C.-P. Lee and C.-b. Lin. Large-scale linear rankSVM. *Neural computation*, 26(4):781–817, 2014.

[13] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui. GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation. KDD '14, pages 831–840. ACM, 2014.

[14] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera. Personalized tour recommendation based on user interests and points of interest visit durations. IJCAI '15, 2015.

[15] Q. Liu, S. Wu, L. Wang, and T. Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. AAAI '16, 2016.

[16] Y. Liu, W. Wei, A. Sun, and C. Miao. Exploiting geographical neighborhood characteristics for location recommendation. CIKM '14, pages 739–748. ACM, 2014.

[17] E. H.-C. Lu, C.-Y. Chen, and V. S. Tseng. Personalized trip recommendation with multiple constraints by mining user check-in behaviors. SIGSPATIAL '12, pages 209–218. ACM, 2012.

[18] X. Lu, C. Wang, J.-M. Yang, Y. Pang, and L. Zhang. Photo2Trip: Generating travel routes from geo-tagged photos for trip planning. MM '10, pages 143–152. ACM, 2010.

[19] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Dover Publications, 1998.

[20] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. WWW '10, pages 811–820. ACM, 2010.

[21] Y. Shi, P. Serdyukov, A. Hanjalic, and M. Larson. Personalized landmark recommendation based on geotags from photo sharing sites. ICWSM '11, 2011.

[22] R. W. Sinnott. Virtues of the haversine. *Sky and telescope*, 68(2):159, 1984.

[23] B. Thomee, B. Elizalde, D. A. Shamma, K. Ni, G. Friedland, D. Poland, D. Borth, and L.-J. Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[24] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann. Time-aware point-of-interest recommendation. SIGIR '13, pages 363–372. ACM, 2013.

[25] Q. Yuan, G. Cong, and A. Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. CIKM '14, pages 659–668. ACM, 2014.

[26] W. Zhang and J. Wang. Location and time aware social collaborative retrieval for new successive point-of-interest recommendation. CIKM '15, pages 1221–1230. ACM, 2015.

[27] Y. Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):29, 2015.

[28] Y. Zheng, L. Capra, O. Wolfson, and H. Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology*, 5(3):38, 2014.

[29] Y.-T. Zheng, Z.-J. Zha, and T.-S. Chua. Mining travel patterns from geotagged photos. *ACM Transactions on Intelligent Systems and Technology*, 3(3):56:1–56:18, May 2012.

A POI Features for Ranking

Table 5: Features of POI p used in rankSVM given query (p_s, p_e, L)

Feature	Description
`category`	one-hot encoding of the category of p
`neighbourhood`	one-hot encoding of the POI cluster that p resides in
`popularity`	logarithm of POI popularity of p
`nVisit`	logarithm of the total number of visit by all users at p
`avgDuration`	logarithm of the average duration at p
`trajLen`	trajectory length L, i.e., the number of POIs required
`sameCatStart`	1 if the category of p is the same as that of p_s, -1 otherwise
`sameCatEnd`	1 if the category of p is the same as that of p_e, -1 otherwise
`sameNeighbourhoodStart`	1 if p resides in the same POI cluster as p_s, -1 otherwise
`sameNeighbourhoodEnd`	1 if p resides in the same POI cluster as p_e, -1 otherwise
`distStart`	distance between p and p_s, calculated using the Haversine formula
`distEnd`	distance between p and p_e, calculated using the Haversine formula
`diffPopStart`	real-valued difference in POI popularity of p from that of p_s
`diffPopEnd`	real-valued difference in POI popularity of p from that of p_e
`diffNVisitStart`	real-valued difference in the total number of visit at p from that at p_s
`diffNVisitEnd`	real-valued difference in the total number of visit at p from that at p_e
`diffDurationStart`	real-valued difference in average duration at p from that at p_s
`diffDurationEnd`	real-valued difference in average duration at p from that at p_e

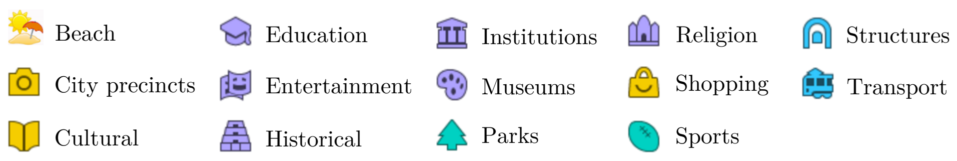


Figure 5: POI Categories

We described an algorithm to recommend trajectories based on ranking POIs (PoiRank) in Section 3.1, the features used to rank POIs are POI and query specific, as described in Table 5.

Categories of POIs in all of the five trajectory datasets are show in Figure 5. The distribution of POI popularity, the number of visit and average visit duration are shown in Figure 6.

To rank POIs, features described in Table 5 are scaled to range $[-1.0, 1.0]$ using the same approach as that employed by libsvm (`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`), i.e., fitting a linear function $f(x) = ax + b$ for feature x such that the maximum value of x maps to 1.0 and the minimum value maps to -1.0.

B Transition Probabilities

Table 6: POI features used to factorise POI-POI transition probabilities

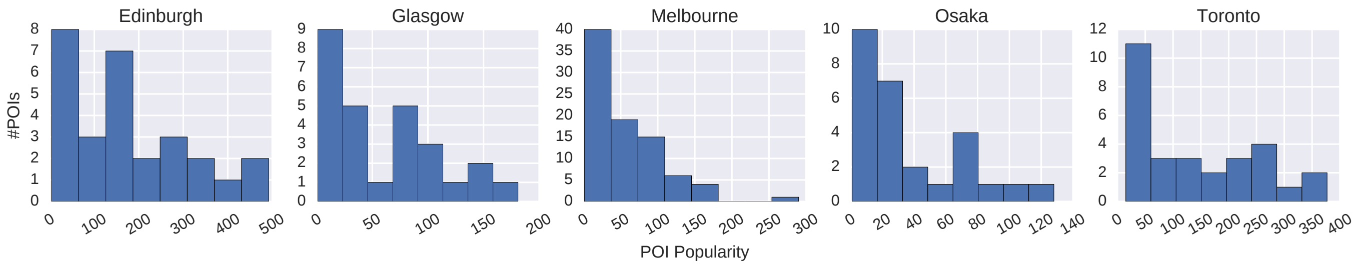
Feature	Description
`category`	category of POI
`neighbourhood`	the cluster that a POI resides in
`popularity`	(discretised) popularity of POI
`nVisit`	(discretised) total number of visit at POI
`avgDuration`	(discretised) average duration at POI

We compute the POI-POI transition matrix by factorising transition probabilities from POI p_i to POI p_j as a product of transition probabilities between pairs of individual POI features, which are shown in Table 6.

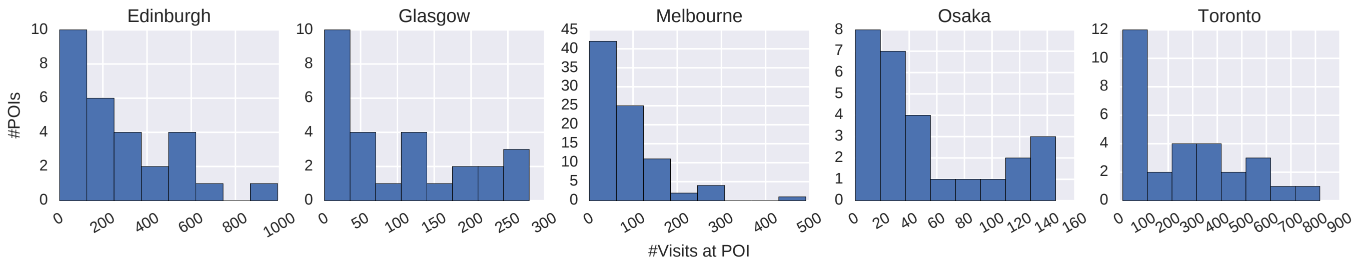
POI Features are discretised as described in Section 2 and transition matrices of individual features are computed using maximum likelihood estimation, i.e., counting the number of transitions for each pair of features then normalising each row, taking care of zeros by adding a small number ϵ [1] to each count before normalisation. Figure 7 visualises the transition matrices for individual POI features in Melbourne.

The POI-POI transition matrix is computed by taking the Kronecker product of the transition matrices for the individual features, and then updating it with the following constraints:

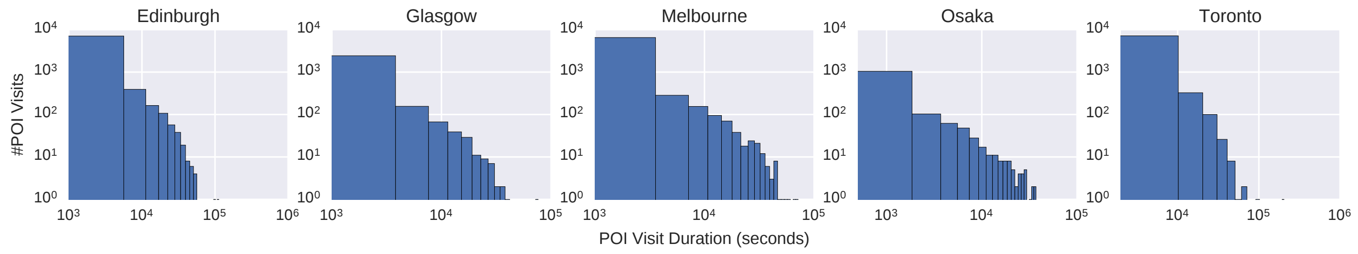
[1]In our experiments, $\epsilon = 1$.



(a) Distribution of POI popularity



(b) Distribution of the number of visit at POI



(c) Distribution of the visit duration at POI

Figure 6: Distribution of POI popularity, the number of visit and visit duration

- Firstly, we disallow self transitions by setting probability of (p_i to p_i) to zero.
- Secondly, when a group of POIs have identical (discretised) features (say a group with M POIs), we distribute the probability uniformly among POIs in the group, in particular, the incoming (unnormalised) transition probability (say, P_{in}) of the group computed by taking the Kronecker product is divided uniformly among POIs in the group (i.e., $\frac{P_{in}}{M}$), which is equivalent to choose a POI in the group uniformly at random. Moreover, the outgoing (unnormalised) transition probability of each POI is the same as that of the group, since in this case, *the transition from any POI in the group to one outside the group represents an outgoing transition from that group*. In addition, the self-loop transition of the group represents transitions from a POI in the group to other POIs ($M - 1$ POIs) in the same group, *similar to the outgoing case*, the (unnormalised) self-loop transition probability (say P_o) is divided uniformly (i.e., $\frac{P_o}{M-1}$), which corresponds to choose a transition (from p_i) among all transitions to the other $M - 1$ POIs (exclude self-loop p_i to p_i) in that group uniformly at random.
- Lastly, we remove feature combinations that has no POI in dataset and normalise each row of the (unnormalised) POI-POI transition matrix to form a valid probability distribution for each POI.

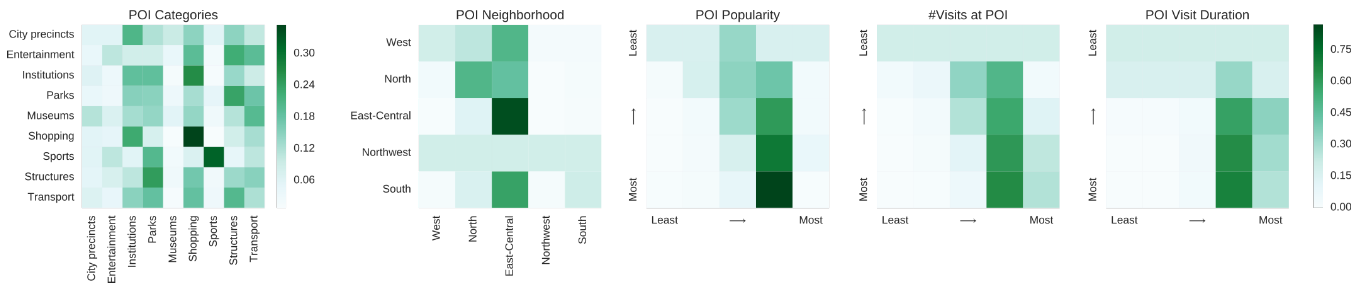


Figure 7: Transition matrices for five POI features: POI category, neighbourhood, popularity, number of visits, and visit duration. These statistics are from the Melbourne dataset.

C Experiment

C.1 Dataset

Trajectories used in experiment (Section 4) are extracted using geo-tagged photos in the Yahoo! Flickr Creative Commons 100M (a.k.a. YFCC100M) dataset [23] as well as the Wikipedia web-pages of points-of-interest (POI). Photos are mapped to POIs according to their distances calculated using the Haversine formula [22], the time a user arrived a POI is approximated by the time the first photo taken by the user at that POI, similarly, the time a user left a POI is approximated by the time the last photo taken by the user at that POI. Furthermore, sequence of POI visits by a specific user are divided into several pieces according to the time gap between consecutive POI visits, and the POI visits in each piece are connected in temporal order to form a trajectory [5, 14].

C.2 Parameters

We use a 0.5 trade-off parameter for PersTour and PersTour-L, found to be the best weighting in [14]. The regularisation parameter C in rankSVM is 10.0. The trade-off parameter α in Rank+Markov and Rank+MarkovPath is tuned using cross validation. In particular, we split trajectories with more than 2 POIs in a dataset into two (roughly) equal parts, and use the first part (i.e., validation set) to tune α (i.e., searching value of α such that Rank+Markov achieves the best performance on validation set, in terms of the mean of pairs-F_1 scores from leave-one-out cross validation), then test on the second part (leave-one-out cross validation) using the tuned α, and vice verse.

C.3 Implementation

We employ the rankSVM implementation in libsvmtools (`https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/`). Integer linear programming (ILP) are solved using Gurobi Optimizer (`http://www.gurobi.com/`) and lp_solve (`http://lpsolve.sourceforge.net/`). Dataset and code for this work are available in repository `https://bitbucket.org/d-chen/tour-cikm16`.

C.4 Performance metric

A commonly used metric for evaluating POI and trajectory recommendation is the F_1 score on points [14], Let $\mathcal{T}$ be the trajectory that was visited in the real world, and $\hat{\mathcal{T}}$ be the recommended trajectory, $\mathcal{P}_{\mathcal{T}}$ be the set of POIs visited in $\mathcal{T}$, and $\mathcal{P}_{\hat{\mathcal{T}}}$ be the set of POIs visited in $\hat{\mathcal{T}}$, F_1 score on points is the harmonic mean of precision and recall of POIs in trajectory,

$$F_1 = \frac{2P_{\text{POINT}}R_{\text{POINT}}}{P_{\text{POINT}} + R_{\text{POINT}}}, \text{ where } P_{\text{POINT}} = \frac{|\mathcal{P}_{\mathcal{T}} \cap \mathcal{P}_{\hat{\mathcal{T}}}|}{|\hat{\mathcal{T}}|} \text{ and } R_{\text{POINT}} = \frac{|\mathcal{P}_{\mathcal{T}} \cap \mathcal{P}_{\hat{\mathcal{T}}}|}{|\mathcal{T}|}.$$

A perfect F_1 (i.e., $\text{F}_1 = 1$) means the POIs in the recommended trajectory are exactly the same set of POIs as those in the ground truth, and $\text{F}_1 = 0$ means that none of the POIs in the real trajectory was recommended.

While F_1 score on points is good at measuring whether POIs are correctly recommended, it ignores the visiting order between POIs. Pairs-F_1 takes into account both the point identity and the visiting orders in a trajectory. This is done by measuring the F_1 score of every pair of ordered POIs, whether they are adjacent or not in trajectory,

$$\text{pairs-F}_1 = \frac{2P_{\text{PAIR}}R_{\text{PAIR}}}{P_{\text{PAIR}} + R_{\text{PAIR}}}, \text{ where } P_{\text{PAIR}} = \frac{N_c}{|\hat{\mathcal{T}}|(|\hat{\mathcal{T}}| - 1)/2} \text{ and } R_{\text{PAIR}} = \frac{N_c}{|\mathcal{T}|(|\mathcal{T}| - 1)/2},$$

and N_c [2] is the number of ordered POI pairs (p_j, p_k) that appear in both the ground-truth and the recommended trajectories,

$$(p_j \prec_{\mathcal{T}} p_k \ \wedge \ p_j \prec_{\hat{\mathcal{T}}} p_k) \ \vee \ (p_j \succ_{\mathcal{T}} p_k \ \wedge \ p_j \succ_{\hat{\mathcal{T}}} p_k),$$

with $p_j \neq p_k, \ p_j, p_k \in \mathcal{P}_{\mathcal{T}} \cap \mathcal{P}_{\hat{\mathcal{T}}}, \ 1 \leq j \neq k \leq |\mathcal{T}|$. Here $p_j \prec_{\mathcal{T}} p_k$ denotes POI p_j was visited before POI p_k in trajectory $\mathcal{T}$ and $p_j \succ_{\mathcal{T}} p_k$ denotes p_j was visited after p_k in $\mathcal{T}$.

Pairs-F_1 takes values between 0 and 1. A perfect pairs-F_1 (1.0) is achieved if and only if both the POIs and their visiting orders in the recommended trajectory are exactly the same as those in the ground truth. Pairs-$\text{F}_1 = 0$ means none of the recommended POI pairs was actually visited in the real trajectory.

Performance data reported in Table 3 and Table 4 are the mean and standard deviation of instances successfully recommended by all methods shown in Table 2.

[2] We define pairs-$\text{F}_1 = 0$ when $N_c = 0$.