Learning Points and Routes to Recommend Trajectories

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 as a tour to travellers, 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 without specialised treatment of spatial, temporal or social information. 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 popular routes. In addition, a probabilistic model and a Structured Support Vector Machine are used to combine the results of POI ranking as well as the POI-POI transitions. To measure performance, such that we respect the order of visits, we propose a new F_1 score on pairs of POIs. Empirical results on trajectory datasets show that our approach improves over previous cutting edge attempts, which demonstrates 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, the local search and discovery service, has 50 million monthly users who have made 8 billion check-ins [13] and Flickr, the online photo-sharing site, hosts over 2 billion geo-tagged public photos [12]. Such large amounts of travel data provide new opportunities for better travel planning traditionally done with written travel guides, for example, choosing and ranking locations for a variety of activities from dining to recreation, and potential new solutions to orienteering and routing problems. 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.

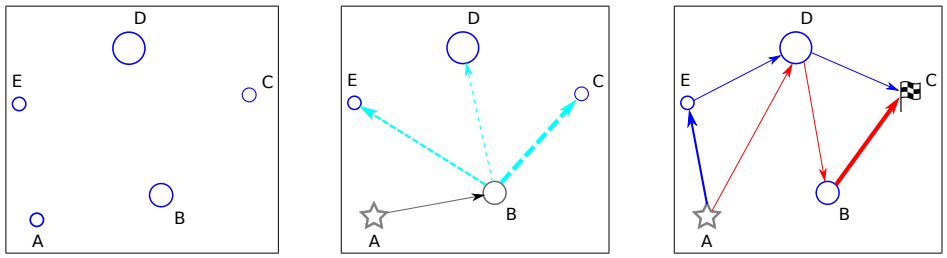
There are several settings of recommendation problems for locations and routes: POI recommendation, next location recommendation, and trajectory recommendation. This is described in more detail in Section 2, and we refer the reader to a number of recent surveys [1, 43, 44] for general overviews of the area. We note, however, that two desired qualities are still missing from the current solutions to trajectory recommendation. The first is 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, social interactions, as they tend to get specialised and separate treatments. This work aims to address these challenges.

We propose a novel way to learn point and route preferences jointly. Our solution consists of several parts. We propose a learning to rank formulation to capture the preference of POIs given the starting and ending points of a trajectory (Section 3). We model pair-wise transition patterns between POIs by factorising the transition likelihoods along five different types of location properties, which alleviates data sparsity between each pair of POIs and allows POIs of the same type and neighbourhood to share parameters. We combine the results of point and transition ranking using Markov chain inference and route planning techniques (Section 4). We further propose a novel way to jointly learn point and transition scores with a Structured Support Vector Machine (StructuredSVM). In Section 5, we evaluate the proposed algorithms on photo trajectories in five different cities, with a traditional metric F_1 on POIs as well as a novel pairs-F_1 metric specifically designed for trajectories.

The main contributions of this work are as follows:

- We propose a novel algorithm to jointly optimise point preference and route plan. We find that the learning-based approaches generally out-perform heuristic route recommendation [24]. Incorporating transitions to POI ranking improves the pairs-F_1 metric for correct sequences of POIs, and that routing algorithms that disallow sub-tours generally out-perform Markov chain methods.
- Our approach is feature-driven and learns from past behaviour without having to design specialised treat-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235



(a) POI Recommendation (b) Next Location Recommendation (c) Trajectory Recommendation

Figure 1: Three settings of trajectory recommendation problems: (a) POI recommendation, (b) next location recommendation, and (c) trajectory recommendation. Node size: POI score; edge width: transition score between pairs of POIs; grey: input query; star: starting location; flag: ending location. See the first four paragraphs of Section 2 for details.

ment 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 [24], 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, which has the property of being between 0 and 1, and being 1 if and only if the recommended trajectory is exactly the same as the ground truth. Our benchmark data and results are publicly available.

2. RELATED WORK

We summarise recent work most related to formulating and solving learning problems on assembling routes from points-of-interest (POI), according to the problem setting, the method for ranking places and routes, as well as the use of different features and queries.

Problem settings for travel patterns and trajectory data. There are several settings of recommendation problems for locations and routes, as illustrated in Figure 1. The first setting can be called points-of-interest (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. This can be in discovery mode, such as identifying points-of-interest [45, 22] and includes efficient querying of geographic objects for trips [16] A popular approach is based on the collaborative filtering model for learning user-location affinity [33], with additional ways to incorporate spatial [23, 27], temporal [37, 17, 14], or spatial-temporal [38] information. There is a rich collection of recent literature on this problem [36, 33, 23, 27, 37, 17, 14, 38], including recent surveys [1, 44].

Figure 1(b) illustrates the second setting: next location recommendation [7, 25, 2, 42]. 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 $A \rightarrow B$. It is a variant of POI recommendation except given both the user and locations travelled to date. The solutions to this problem include incorporating Markov chains into collaborative filtering [32, 7, 42], quantifying tourist traffic flow between points-of-interest [46], formulating a binary decision or ranking problem [2], and with sequence models such as recurrent neural networks [25].

This paper considers the final setting: route recommendation, as illustrated in 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 [29, 24, 28], or formulates an optimisation problem that is not informed or evaluated by behaviour history [15, 5]. Joint learning of location preferences and routes remains an open problem. Our work is in this category. We formulate a learning problem to score the whole trajectory, taking into account individual POI properties and relationships among different POIs.

Methods for ranking locations and trajectories. The first is collaborative filtering, or matrix factorisation family of techniques applied to places and trajectories [33, 7, 42]. The second type regards route recommendation as a planning problem [15, 24]. The TripBuilder system [4] first solves a maximum coverage problem for user preference and then solves TSP for routing. The collaborative filtering approaches rank POI but do not take into account the sequence that a trip is taken. On the other hand, the planning approaches assume a fixed objective function that is not directly optimised to predict user behaviour. There have been a few approaches that jointly consider POI preferences and routes [28, 20, 5]. We propose a new approach that combines different POI features and POI-POI transition features, while learning the relevant weights from data. We also provide a comprehensive benchmark on how node and edge features contribute to the final trajectory ranking.

A diverse set of available information, such as geographic, time, user properties, and subjective opinions. Approaches for modelling space include geolocation-informed matrix factorisation [23], spatial topic models [18], neighbourhood information [27], exploiting sequential information with additive Markov chain [40], and enriching location with the information for venues [9, 10]. Important variants to consider for modelling time include travel time [14], constructing time-aware routes [37, 17], time-of-day and day-

Figure 2: Transition matrices for five POI features: POI categories, neighborhood, popularity, number of visits, and visit duration. These statistics are from the Melbourne dataset. See Section 3.2 for descriptions.

of-week [5], POI availability and uncertainty in travelling time [39]. There are a number of approaches that jointly consider space and time [38, 42], such as modelling the correlation between check-in time and location [14]. Inferring user attributes and preferences has been an important consideration [26]. Chen et al [6] recommend places to travellers based on user demographics and travel group types (e.g., couple, family and friends) from online photos. Ference et al [11] tailors the recommendation for out of town users. Subjective opinion is an important information source for decision making, in addition to past behaviours. Zhang et al [41] use written reviews for POI recommendation. The novel work from Quercia et al [31] crowd-source judgements about the beauty, quietness and happiness of places, and then constructs routes with these subjective criteria.

In this work, we focus on formulating learning problems to jointly rank POIs and routes. We use a minimum but commonly available set of features, such as neighbourhood, popularity, and venue information. Our learning algorithm can easily incorporate other features as long as they can be encoded in unary or pair-wise form. Gathering additional features via data linking has challenging research issues on its own, and is out of the scope for this paper.

3. POI, QUERY AND PATH

We aim to recommend a particular tour such that the user will visit a sequence of points-of-interests (POIs), denoted $p_1, \ldots, p_L$, that maximises utility. We are given the desired start ($p_1 = p_s$) and end point ($p_L = p_e$), and an associated number L of POIs desired, from which we propose a trajectory through the city. In Figure 1(c), an example tour is shown in blue, which starts at the POI denoted as a grey star, visits two intermediate POIs, and terminates on the fourth POI denoted as a flag. The tour of length 4 can be modelled as a sequence of directed edges in a graph containing POIs in the city as nodes.

The training data consists of a set of tours of varying length in a particular city. We assume that all POIs $\mathcal{P}$ in the city are visited at least once, and hence can construct a graph with POIs as nodes and potentially multiple directed edges between each pair of nodes. We extract the category, popularity, total number of visits and average visit duration for each POI.

A naive approach would be to recommend the trajectory based on the popularity (number of distinct visitors) [8] 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 request is to adjust k to match the desired length.

3.1 Ranking using the origin and destination

In addition to popularity, we also can rank the candidate POIs based on the other three POI specific features (category, total visits and average duration). We can learn a ranking of POIs using rankSVM with linear kernel and $L2$ loss [21],

$$\min_{\mathbf{w_r}} \frac{1}{2}\mathbf{w_r}^T\mathbf{w_r} + C_r \sum_{(p_i,p_j)\in\mathcal{P}} \max\left(0, 1 - \mathbf{w_r}^T(\mathbf{f}_{p_i} - \mathbf{f}_{p_j})\right)^2$$

where $\mathcal{P}$ is the set of POIs to rank, $\mathbf{w_r}$ is a vector of parameters, $C_r > 0$ is the regularisation parameter and $\mathbf{f}_{p_i}$ is the feature vector for POI p_i.

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, using the query (p_s, p_e, L) we can construct new features by contrasting candidate POIs with p_s and p_e.

For each of the POI features (category, total visits and average duration), we construct two new features by taking the difference of the feature in POI p with (p_s, p_e) respectively. For the category, we set the feature to 1 when their categories are the same and -1 otherwise. For popularity, total visits and visit duration, we take the real valued difference. The distance from POI p to p_s (and p_e) is computed using the Haversine formula [34]. Lastly we also include the required length L of the trajectory as a feature for rankSVM. The frequency distribution of POI popularity, number of visits and visit duration of five datasets are shown in Figure 5.

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. We create another algorithm, PoiRank, to recommend trajectory by ranking POIs utilising both POI and query specific features described above. PoiRank takes the top ranked $L-2$ POIs and connects them in sequence of the ranks.

3.2 Transition probabilities

In addition to information about each individual POI, a tour recommendation system would benefit from capturing the likelihood of transitioning between different POIs. One option would be to directly model the probability of going from one POI to another, but this has several weaknesses: Such a model would be unable to handle a new POI (one

that has not yet been visited). Furthermore, even if we restrict ourselves to known POIs, there may be many locations which are rarely visited, leading to significant challenges in estimating the probabilities from empirical data.

We model POI transitions using a Markov chain with discrete factored states. One difficulty of estimating the Markov chain is data sparsity, i.e., not enough transitions have been observed between every pair of POIs. We factorised the transition probability from POI p_i to POI p_j as a product of transition probabilities between pairs of individual POI features (listed in Section 3.1). POIs are grouped into 5 clusters using K-means according to their geographical locations to reflect their neighbourhood relations. We directly model the transition between the category and neighbourhood of each POI as the conditional probability. The popularity, total number of visits and the average visit duration are discretised by binning them uniformly into 5 discrete intervals on the log scale. Figure 2 visualises the transition matrices for individual POI features in Melbourne.

We compute the transition probabilities of the above individual POI features 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, which results in a transition matrix for each of the above POI features.

Assuming independence between these features, the transition probability $P(p_j|p_i)$ from p_i to p_j can be defined as the product of the transition probabilities of each individual feature (and appropriately normalised), with two additional constraints. First we disallow self transitions by setting the probability of (p_i to p_i) to zero. Second we need to deal with the possibility that multiple POIs may share exactly the same feature vector. When a group of POIs have identical (discretised) features, we distribute the probability uniformly among the POIs in the group. The POI-POI transition matrix can be efficiently computed by taking the Kronecker product of the transition matrices for the individual features and then updating it based on the constraints described above.

Given the POI-POI transition probabilities, we recommend a trajectory with respect to query (p_s, p_e, L) by maximising the (log) likelihood. We call this approach that only uses the transition probabilities between POIs as MARKOV. The maximum likelihood solution can be found using a variant of the Viterbi algorithm (with emission probabilities ignored), which is shown in Algorithm 1. The entry $A[l, p]$ in score matrix A stores the maximum likelihood associated with the (partial) trajectory that starts from p_s and ends at p with l POI visits, and entry $B[l, p]$ in the backtracking-point matrix B stores the predecessor of p in that (partial) trajectory.

3.3 Avoiding sub-tours

The tours recommended by MARKOV described in Section 3.2 are found using the maximum likelihood approach, and may contain multiple visits to the same POI. This is because the best path (or walk) from Viterbi decoding may have tottering (where the Markov chain transitions back and forth between two states), or 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 specifying additional constraints when recommending trajectories.

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

Algorithm 1 MARKOV: recommend trajectory with POI transitions

Input: $\mathcal{P}, p_s, p_e, L$
Output: Trajectory $\mathcal{T} = (p_s, \cdots, p_e)$ with L POIs
Initialise score matrix A and backtracking pointer B
for $p \in \mathcal{P}$ **do**
  $A[2, p] = \log P(p|p_s)$
  $B[2, p] = p_s$
end for
for $l = 2$ to $L - 1$ **do**
  for $p \in \mathcal{P}$ **do**
    $A[l+1, p] = \max_{p' \in \mathcal{P}} \{A[l, p'] + \log P(p|p')\}$
    $B[l+1, p] = \operatorname{argmax}_{p' \in \mathcal{P}} \{A[l, p'] + \log P(p|p')\}$
  end for
end for
$\mathcal{T} = \{p_e\}$, $l = L$, $p = \mathcal{T}.first$
repeat
  Prepend $B[l, p]$ to $\mathcal{T}$
  $l = l - 1$, $p = \mathcal{T}.first$
until $l < 2$
return $\mathcal{T}$

In particular, we find the best trajectory using an integer linear program with sub-tour elimination constraints adapted from the Travelling Salesman Problem [30]. We call our method that uses the transition matrix to recommend paths that do not have circular sub-tours MARKOVPATH.

Given a set of POIs $\mathcal{P}$, the POI-POI transition matrix and a query (p_s, p_e, L), we recommend a trajectory by solving the following integer linear program:

$$\max \sum_{i=1}^{N-1} \sum_{j=2}^{N} x_{ij} \log P(p_j|p_i)$$

$$s.t.\ x_{ij} \in \{0, 1\},\ \forall i, j = 1, \cdots, N \quad (1)$$

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1 \quad (2)$$

$$\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \leq 1,\ \forall k = 2, \cdots, N-1 \quad (3)$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} x_{ij} = L - 1 \quad (4)$$

$$u_i - u_j + 1 \leq (N-1)(1 - x_{ij}),\ \forall i, j = 2, \cdots, N \quad (5)$$

where $N = |\mathcal{P}|$ is the number of POIs and x_{ij} is a binary decision variable which determines whether transition from POI p_i to POI p_j occurred in the recommended trajectory. For brevity, we assume x_{i1} and x_{1j} represent the incoming and outgoing transitions of p_s, similarly, x_{iN} and x_{Nj} correspond to the incoming and outgoing transitions of p_e. Constraint (2) restricts that only one outgoing (incoming) transition for p_s (p_e) is permitted, i.e., the recommended trajectory should start from p_s and end at p_e. Constraint (3) restricts that any POI could be visited at most once and constraint (4) restricts that only $L - 1$ transitions between POIs are permitted, i.e., the number of visited POIs should be exactly L (including p_s and p_e). The last constraint re-

stricts that no sub-tours are permitted in the recommended trajectory.

4. TOUR RECOMMENDATION

Recall that in Section 3.1 we described how to recommend trajectories by ranking points, i.e., PoiRank and ones that consider transitions between POIs, i.e., Markov and MarkovPath. In this section, we propose three approaches to jointly optimise the recommendation with both POI preferences and route plans. A summary of the various trajectory recommendation approaches can be found in Table 2.

4.1 POI ranking and transitions

To recommend the *most likely* trajectory with respect to a query, we want to combine the ranking of POIs with the transition probabilities, i.e., we want to find a trajectory that maximise the points ranking of its POIs as well as its likelihood at the same time.

Firstly, we transform the ranking scores of POIs with respect to query $q = (p_s, p_e, L)$ to a probability distribution using the softmax function [3],

$$P_R(p_j|q) = \frac{e^{R(p_j)}}{\sum_j e^{R(p_j)}} \quad (6)$$

where $R(p_j)$ is the ranking score of POI p_j from rankSVM with respect to query q.

One heuristic to find a trajectory that simultaneously maximise the product of ranking probabilities of its POIs (with respect to query q) and its likelihood is optimising the following objective:

$$\underset{\mathcal{T}\in\mathcal{P}^L}{\operatorname{argmax}}\ \alpha \prod_{k=1}^{L} P_R(p_{j_k}|q) + (1-\alpha) \prod_{k=1}^{L-1} P(p_{j_{k+1}}|p_{j_k})$$

such that $p_{j_1} = p_s,\ p_{j_L} = p_e$ and $p_{j_k} \in \mathcal{P}, 1 \le k \le L$. $\mathcal{T} = (p_{j_1}, \ldots, p_{j_L})$ is any possible trajectory and $0 \le \alpha \le 1$ is a parameter to trade-off the importance between the ranking of POIs and the POI-POI transitions in the recommended trajectory. Similar to the Markov algorithm mentioned above, the best path (or walk) can be found using the Viterbi algorithm with both node and transition scores.

$$A[l+1,p] = \max_{p' \in \mathcal{P}} \{A[l,p'] + \alpha \log P_R(p|q) + (1-\alpha) \log P(p|p')\} \quad (7)$$

$$B[l+1,p] = \underset{p' \in \mathcal{P}}{\operatorname{argmax}} \{A[l,p'] + \alpha \log P_R(p|q) + (1-\alpha) \log P(p|p')\} \quad (8)$$

where A the score matrix and entry $A[l,p]$ stores the maximum value that associated with the (partial) trajectory, which 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 $B[L, p_e]$. We call this approach that uses both the ranking of POIs and POI-POI transitions Rank+Markov, with the pseudo code shown in Algorithm 2.

Similar to the Markov algorithm described in Section 3.2, sub-tours (e.g., tottering) may appear in trajectories recommended by Rank+Markov. To eliminate sub-tours, we can optimise the following objective using integer linear programming with the same constraints as those in MarkovPath algorithm described in Section 3.3.

$$\max \sum_{i=1}^{N-1} \sum_{j=2}^{N} x_{ij} (\alpha \log P_R(p_j|q) + (1-\alpha) \log P(p_j|p_i))$$

We call this algorithm Rank+MarkovPath in experiments. Hyperparameter α is tuned using cross validation for both Rank+Markov and Rank+MarkovPath.

Algorithm 2 Rank+Markov: recommend trajectory with both POI ranking and transition

Input: $\mathcal{P}, p_s, p_e, L$
Output: Trajectory $\mathcal{T} = (p_s, \cdots, p_e)$ with L POIs
Initialise score matrix A and backtracking pointers B
for $p \in \mathcal{P}$ **do**
  $A[2,p] = \alpha(\log P_R(p_s|q) + \log P_R(p|q)) + (1-\alpha) \log P(p|p_s)$
  $B[2,p] = p_s$
end for
for $l = 2$ to $L-1$ **do**
  for $p \in \mathcal{P}$ **do**
    Compute $A[l+1,p]$ using Equation (7)
    Compute $B[l+1,p]$ using Equation (8)
  end for
end for
$\mathcal{T} = \{p_e\}$, $l = L$, $p = \mathcal{T}.first$
repeat
  Prepend $B[l,p]$ to $\mathcal{T}$
  $l = l-1$, $p = \mathcal{T}.first$
until $l < 2$
return $\mathcal{T}$

4.2 Structured SVM

As trajectory is a sequence of POI visits, it is natural to model the recommended trajectory $\mathcal{T}$ with respect to query $q = (p_s, p_e, L)$ as a chain of L variables, where each discrete variable has $|\mathcal{P}|$ states. Structured prediction incorporates both the features of variables (unary features) and the features of interactions between neighbouring variables (pairwise features) to make a prediction,

$$\mathcal{T}^* = \underset{\mathcal{T}\in\mathcal{P}^L}{\operatorname{argmax}} \sum_{j=1}^{L} \mathbf{w_u}^T \phi_j(q, \mathcal{T}_j) + \sum_{j=1}^{L-1} \mathbf{w_p}^T \phi_{j,j+1}(q, \mathcal{T}_j, \mathcal{T}_{j+1})$$

where ϕ_j are the unary features of the j-th variable and $\phi_{j,j+1}$ are the pairwise features between the j-th and $(j+1)$-th variables, $\mathbf{w_u}$ and $\mathbf{w_p}$ are the parameters of unary and pairwise features respectively.

The unary and pairwise features are constructed from the POI ranking and POI-POI transitions. In particular, unary features are defined as ranking probabilities (Equation 6). Recall that we have a query consisting of start (p_s) and end (p_e) locations, which we model as a 1-of-K encoding in the unary features. The pairwise features are defined from the transition probabilities $P(p_j|p_i)$ described in Section 3.2. This method is called StructuredSVM in the experiments.

To estimate the parameters $\mathbf{w_u}$ and $\mathbf{w_p}$, we train a Structured Support Vector Machine using the 1-slack formula-

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

tion [19],

$$\min_{\mathbf{w},\xi\geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$
$$s.t. \quad \forall \left(\hat{\mathcal{T}}^{(1)}, \cdots, \hat{\mathcal{T}}^{(N)}\right) \in \mathscr{T}^N :$$
$$\frac{1}{N}\mathbf{w}^T \sum_{i=1}^{N} \delta\left(\hat{\mathcal{T}}^{(i)}\right) \geq \frac{1}{N}\sum_{i=1}^{N} \Delta\left(\mathcal{T}^{(i)}, \hat{\mathcal{T}}^{(i)}\right) - \xi$$

where $\mathbf{w} = [{\mathbf{w_u}}^T, {\mathbf{w_p}}^T]^T$ is the parameter vector, $\mathcal{T}^{(i)}$ and $\hat{\mathcal{T}}^{(i)}$ are the i-th trajectory in training set and its corresponding recommendation respectively. N is the training set size, C is the regularisation parameter, ξ is the slack variable, and

$$\delta\left(\hat{\mathcal{T}}^{(i)}\right) = \Psi\left(q^{(i)}, \mathcal{T}^{(i)}\right) - \Psi\left(q^{(i)}, \hat{\mathcal{T}}^{(i)}\right)$$

where $q^{(i)}$ the query corresponds to the i-th trajectory in training set, $\Psi\left(q^{(i)}, \mathcal{T}^{(i)}\right)$ is the joint feature vector which is a composite of unary and pairwise features of the i-th trajectory, $\Delta\left(\mathcal{T}^{(i)}, \hat{\mathcal{T}}^{(i)}\right)$ is the loss associated with the i-th trajectory and its corresponding recommendation, and Hamming loss is used in this work.

5. EXPERIMENTAL RESULTS

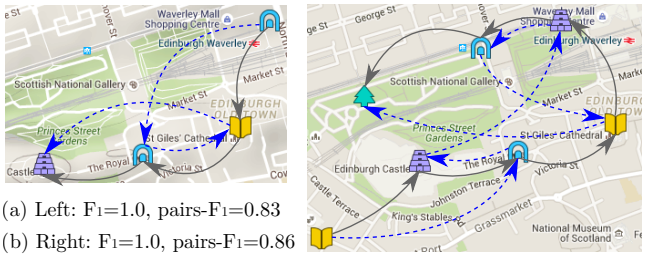
5.1 Photo trajectories from five cities

We experiment on trajectory datasets from five cities, namely, Edinburgh, Glasgow, Osaka, Toronto and Melbourne. The statistics of these trajectory datasets are described in Table 1. The first four datasets are provided by Lim et al. [24] and the Melbourne dataset is built using the same approach introduced in earlier work [8, 24] and briefly described below.

Trajectories are extracted using geo-tagged photos in the Yahoo! Flickr Creative Commons 100M (a.k.a. YFCC100M) dataset [35] as well as the Wikipedia web-pages of points-of-interests (POIs). Photos are mapped to POIs according to their distances calculated using the Haversine formula [34], 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.

5.2 Performance metrics

A commonly used metric for POI recommendation and evaluating trajectories is the F_1 score on points. 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 $\hat{\mathcal{T}}$, we compute trajectory F_1 as the harmonic mean between the point-wise precision and recall.



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

Figure 3: Two illustrative examples for F_1 vs pairs-F_1 as evaluation metric for trajectories. Solid grey: ground truth trajectories; dashed blue: recommended trajectories. See Section 5.2 for details.

$$F_1 = \frac{2P_{\text{POINT}}R_{\text{POINT}}}{P_{\text{POINT}} + R_{\text{POINT}}}$$

where

$$P_{\text{POINT}} = \frac{|\mathcal{P}_\mathcal{T} \cap \mathcal{P}_{\hat{\mathcal{T}}}|}{|\hat{\mathcal{T}}|}, R_{\text{POINT}} = \frac{|\mathcal{P}_\mathcal{T} \cap \mathcal{P}_{\hat{\mathcal{T}}}|}{|\mathcal{T}|}$$

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

While trajectory F_1 is good at measuring whether POIs are correctly recommended, it ignores the visiting order between POIs. 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 listed in Table 2. Both examples have a perfect F_1 score, but not a perfect pairs-F_1 score due to the difference in POI sequencing. We propose a new metric pairs-F_1 that 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.

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

where

$$P_{\text{PAIR}} = \frac{N_c}{|\hat{\mathcal{T}}|(|\hat{\mathcal{T}}|-1)/2}, R_{\text{PAIR}} = \frac{N_c}{|\mathcal{T}|(|\mathcal{T}|-1)/2}$$

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

$$(p_j \prec_\mathcal{T} p_k \ \wedge \ p_j \prec_{\hat{\mathcal{T}}} p_k) \ \vee \ (p_j \succ_\mathcal{T} p_k \ \wedge \ p_j \succ_{\hat{\mathcal{T}}} p_k)$$
$$\text{with } p_j \neq p_k, \quad p_j, p_k \in \mathcal{P}_\mathcal{T} \cap \mathcal{P}_{\hat{\mathcal{T}}}, \ j \neq k, \quad 1 \leq j, k \leq |\mathcal{T}|.$$

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

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

	Query	POI	Trans.	No sub-tours	Joint
RANDOM	×	×	×	×	×
PERSTOUR[24]	×	√	×	√	×
PERSTOUR-L	×	√	×	√	×
POIPOPULARITY	×	√	×	×	×
POIRANK	√	√	×	×	×
MARKOV	×	√	√	×	×
MARKOVPATH	×	√	√	√	×
RANK+MARKOV	√	√	√	×	×
RANK+MARKOVPATH	√	√	√	√	×
STRUCTUREDSVM	√	√	√	×	√

the same as those in the ground truth. Pairs-$F_1 = 0$ means none of the recommended POI pairs was actually visited in the real trajectory.

5.3 Experimental setup

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 from recent literature, the 10 variants of approaches are summarised below, with an overview in Table 2.

Baseline approaches including RANDOM which naively chooses POIs uniformly at random (without replacement) from the set of POIs $\mathcal{P} \setminus \{p_s, p_e\}$ to form a trajectory. It does not utilise any features related to POI or query, as shown in Table 2. Among the related approaches, PERSTOUR [24] is the most similar work and it explores both POI features and the sub-tour elimination constraints described in Section 3.3, in addition, the recommended trajectory is constrained by a time budget. One variant of PERSTOUR is applying the trajectory length instead of the time budget to constrain the recommended trajectory, which is denoted as PERSTOUR-L in Table 2. Another baseline is POIPOPULARITY described at the beginning of Section 3, it only uses POI popularity to recommend a trajectory.

Variants of point- and route-ranking approaches including POIRANK (Section 3.1) that utilises POI and query features, and MARKOV (Section 3.2), that recommends trajectories uses only transition probabilities between POIs and its variant MARKOVPATH that incorporates additional constraints to eliminate sub-tours. Both RANK+MARKOV and RANK+MARKOVPATH (Section 4.1) utilise POI and query features as well as transition probabilities, with the latter uses additional sub-tour elimination constraints. Lastly, we have the STRUCTUREDSVM (Section 4.2) that not only utilises POI and query features as well as transition probabilities but also jointly optimises their relative importance.

Parameters of these algorithms. We use a 0.5 trade-off parameter for PERSTOUR and PERSTOUR-L, found to be the best weighting by PERSTOUR [24]. We set the regularisation parameter in rankSVM to 10.0, determined by cross-validation. The trade-off value in RANK+MARKOV and RANK+MARKOVPATH is also 0.5 and the regularisation parameter in STRUCTUREDSVM is 1.0.

5.4 Results

The performance of the baseline algorithm and the POI ranking and routing variants on five trajectories datasets are summarised in Table 3 and Table 4, in terms of trajectory F_1 score and pairs-F_1, respectively.

We can see from Table 3 that MARKOVPATH and RANK+MARKOVPATH outperform their corresponding variants MARKOV and RANK+MARKOV without the path constraints in terms of F1. This demonstrates that eliminating sub-tours improves point recommendation. This is expected, as sub-tours worsen the proportion of correctly recommended POIs since a length constraint is used. On the contrary, most Markov chain entries have better performance in terms of pairs-F_1 which indicates Markov chain approaches generally respect the transition patterns between POIs. The effect of sub-tours on trajectory recommendation can be further observed through the performance of STRUCTUREDSVM. While utilising the same features as RANK+MARKOV and RANK+MARKOVPATH, it is more flexible with automatic tuning of the trade-off of points and routes, resulting in the best performance with respect to pair-F1 in Edinburgh, Osaka and Toronto.

Algorithms based on POI rankings yield strong performance by exploring POI and query specific features. POIRANK improves notably upon POIPOPULARITY and PERSTOUR by leveraging more POI features and query specific features in 3 out of 5 datasets. On the other hand, Algorithm that leverage transition probabilities alone does not perform particularly well. In terms of trajectory F_1, MARKOV can not outperform POIRANK except on Osaka dataset, on which the performance of both algorithms are comparable. In particular, algorithms with ranking information (e.g. RANK+MARKOV and RANK+MARKOVPATH) always out-perform their respective variants with transition information alone (i.e. MARKOV and MARKOVPATH).

POIRANK performs significantly better than MARKOV on three datasets, namely, Glasgow, Melbourne and Toronto. This result may seem strange at first sight because MARKOV modelled the transition patterns and it could leverage this advantage to accomplish better performance in terms of pairs-F_1 given the fact that this metric is measuring the quality of visiting order. However, pairs-F_1 is actually affected heavily by the proportion of correctly recommended POIs in trajectory, once the POIs are correctly predicted, a better pairs-F_1 would be achieved, this supposition can be confirmed by the observation that, on Osaka dataset, MARKOV and POIRANK achieved comparable performance in terms of trajectory F_1, which means the proportion of correctly recommended POIs is comparable, but MARKOV outperforms POIRANK significantly in terms of pairs-F_1, which suggest the transition information is helpful in some situations.

Transition information helps point ranking when their performances are comparable. When the advantage gap of POIRANK over MARKOV is large, i.e. Edinburgh, Glasgow, Melbourne and Toronto in Table 3 and Glasgow, Toronto in Table 4, transition information does not help RANK+MARKOV, which explores both POI ranking and transitions, can not compete with POIRANK in both metrics, but nevertheless brings improvements to MARKOV. However, as the performance of MARKOV approaches or surpasses that of POIRANK, i.e., Osaka in Table 3 and Edinburgh, Melbourne, Osaka in Table 4, RANK+MARKOV outperforms both of them, this observation indicates that transition information can be very helpful when ranking does not performs well enough.

Table 3: Performance comparison on five datasets in terms of trajectory 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 ± 0.124	0.558 ± 0.149	0.621 ± 0.117	0.621 ± 0.128
PersTour[24]	0.656 ± 0.223	$\mathbf{0.802 \pm 0.213}$	0.491 ± 0.211	0.702 ± 0.230	0.720 ± 0.215
PersTour-L	0.651 ± 0.143	0.660 ± 0.102	0.578 ± 0.140	0.691 ± 0.138	0.642 ± 0.112
PoiPopularity	$\mathbf{0.701 \pm 0.160}$	0.745 ± 0.166	0.621 ± 0.136	0.661 ± 0.128	0.679 ± 0.120
PoiRank	$\mathit{0.694 \pm 0.157}$	$\mathit{0.777 \pm 0.171}$	$\mathbf{0.626 \pm 0.137}$	0.679 ± 0.112	$\mathbf{0.748 \pm 0.166}$
Markov	0.629 ± 0.172	0.714 ± 0.168	0.577 ± 0.168	0.679 ± 0.162	0.663 ± 0.157
MarkovPath	0.678 ± 0.148	0.735 ± 0.170	0.596 ± 0.147	0.706 ± 0.154	0.689 ± 0.140
Rank+Markov	0.642 ± 0.171	0.736 ± 0.176	0.598 ± 0.169	0.701 ± 0.171	0.689 ± 0.170
Rank+MarkovPath	0.684 ± 0.151	0.760 ± 0.170	$\mathit{0.625 \pm 0.150}$	$\mathbf{0.719 \pm 0.161}$	0.724 ± 0.152
StructuredSVM	0.659 ± 0.186	0.727 ± 0.173	0.597 ± 0.171	$\mathit{0.715 \pm 0.170}$	$\mathit{0.728 \pm 0.186}$

Table 4: Performance comparison on five datasets in terms of pairs-F_1. 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 ± 0.169	0.249 ± 0.148	0.305 ± 0.145	0.311 ± 0.167
PersTour[24]	0.417 ± 0.343	$\mathbf{0.646 \pm 0.366}$	0.225 ± 0.274	$\mathit{0.491 \pm 0.377}$	0.503 ± 0.353
PersTour-L	0.359 ± 0.207	0.352 ± 0.162	0.268 ± 0.143	0.415 ± 0.243	0.331 ± 0.159
PoiPopularity	$\mathit{0.436 \pm 0.259}$	0.509 ± 0.299	0.316 ± 0.178	0.363 ± 0.195	0.385 ± 0.202
PoiRank	0.424 ± 0.249	$\mathit{0.565 \pm 0.312}$	0.322 ± 0.186	0.376 ± 0.173	$\mathit{0.512 \pm 0.295}$
Markov	0.424 ± 0.238	0.488 ± 0.283	0.297 ± 0.192	0.449 ± 0.262	0.419 ± 0.237
MarkovPath	0.400 ± 0.234	0.492 ± 0.298	0.294 ± 0.187	0.445 ± 0.268	0.407 ± 0.234
Rank+Markov	0.434 ± 0.251	0.540 ± 0.294	$\mathbf{0.357 \pm 0.210}$	0.483 ± 0.277	0.462 ± 0.266
Rank+MarkovPath	0.408 ± 0.239	0.532 ± 0.304	0.331 ± 0.213	0.470 ± 0.284	0.465 ± 0.266
StructuredSVM	$\mathbf{0.440 \pm 0.267}$	0.529 ± 0.283	$\mathit{0.352 \pm 0.213}$	$\mathbf{0.508 \pm 0.292}$	$\mathbf{0.520 \pm 0.311}$

PersTour [24] always performs better than its variant PersTour-L, especially on Glasgow and Toronto datasets. This indicate the time budget constraint is more helpful than length constraint for recommending trajectories. The exception on Melbourne dataset is explained next. It is obvious that algorithms captured a certain type of information as shown in Table 2 outperforms the Random baseline in terms of both performance metrics on all five datasets, except the performance of PersTour[24] on Melbourne dataset which will be interpreted later.

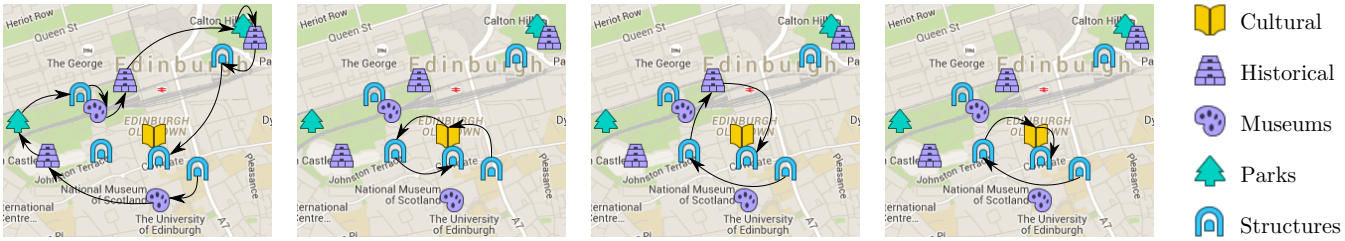
Lastly, we observed that PersTour is surprisingly outperformed by Random baseline on Melbourne dataset in terms of both metrics. It turns out that, on this very dataset, many of the integer linear programming (ILP) problems which PersTour need to solve to get the recommendations are difficult ILP instances. In the leave-one-out evaluation, although we utilised a large scale computing cluster with modern hardware, 12% of evaluations failed due to the ILP solver could not find a feasible solution in 2 hours. Furthermore, a lot of recommendations are produced from sub-optimal solutions of the corresponding ILPs due to the 2 hours timeout setting, these factors lead to the inconsistent performance of PerTour on Melbourne dataset.

5.5 Case study

Figure 4 illustrates a test case in Edinburgh dataset. 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 on 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 using POI specific features (illustrated in Figure 5) as well as query specific features, but with completely different routes. On the other hand, Markov (Figure 4(c)) missed one POI but the recommended route for one of the intermediate 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.

6. CONCLUSION

In this paper, we propose an approach to recommend trajectories by jointly optimising points and routes. This is in contrast to related work which looks at POI recommendation or next location recommendation. Point preferences are learned by ranking according to POI and query features, and transition probabilities between POIs are learned from previous trajectories extracted from social media. We investigate the weaknesses of a naive maximum likelihood sequence approach (which may recommend sub-tours) and propose an improved sequence prediction method. Our feature driven approach allows a natural combination of POI ranks and routes, and we proposed two ways (directly combining them, or learning a structured SVM) for unifying the two models. We argue that one should measure performance also with respect to the visiting order of POIs, and suggest a new pairs-F_1 metric. We empirically evaluated our tour recommendation approaches on five datasets extracted from Flickr photos, and demonstrate that our method improves on prior work, on both the traditional F_1 metric and our proposed performance measure. Our promising results from learning points and routes for trajectory recommendation suggests



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

Figure 4: Different recommendations from algorithm variants. See the main text in Section 5.5 for description.

that research in this domain should consider both information sources simultaneously. Since our machine learning method captures domain knowledge in terms of the features, it does not rely on the fact that the current task uses spatial and social information, therefore it may be more widely applicable to other trajectory recommendation tasks.

7. 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. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] I. Brilhante, J. A. Macedo, F. M. Nardini, R. Perego, and C. Renso. Where shall we go today?: planning touristic tours with tripbuilder. CIKM '13, pages 757–762. ACM, 2013.

[5] C. Chen, D. Zhang, B. Guo, X. Ma, G. Pan, and Z. Wu. TripPlanner: Personalized trip planning leveraging heterogeneous crowdsourced digital footprints. *IEEE Transactions on Intelligent Transportation Systems*, 16(3):1259–1273, 2015.

[6] Y.-Y. Chen, A.-J. Cheng, and W. H. Hsu. Travel recommendation by mining people attributes and travel group types from community-contributed photos. *IEEE Transactions on Multimedia*, 15(6):1283–1295, 2013.

[7] 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.

[8] 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.

[9] R. Deveaud, M. Albakour, C. Macdonald, I. Ounis, et al. On the importance of venue-dependent features for learning to rank contextual suggestions. CIKM '14, pages 1827–1830. ACM, 2014.

[10] R. Deveaud, M. Albakour, C. Macdonald, I. Ounis, et al. Experiments with a venue-centric model for personalised and time-aware venue suggestion. CIKM '15, pages 53–62. ACM, 2015.

[11] G. Ference, M. Ye, and W.-C. Lee. Location recommendation for out-of-town users in location-based social networks. CIKM '13, pages 721–726. ACM, 2013.

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

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

[14] 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.

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

[16] T. Hashem, S. Barua, M. E. Ali, L. Kulik, and E. Tanin. Efficient computation of trips with friends and families. CIKM '15, pages 931–940. ACM, 2015.

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

[18] B. Hu and M. Ester. Spatial topic modeling in online social media for location recommendation. RecSys '13, pages 25–32. ACM, 2013.

[19] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[20] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura. Travel route recommendation using geotags in photo sharing sites. CIKM '10, pages 579–588. ACM, 2010.

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

[22] X. Li, T.-A. N. Pham, G. Cong, Q. Yuan, X.-L. Li, and S. Krishnaswamy. Where you instagram?: Associating your instagram photos with points of interest. CIKM '15, pages 1231–1240. ACM, 2015.

[23] 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.

[24] 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.

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

[26] X. Liu, Y. Liu, K. Aberer, and C. Miao. Personalized point-of-interest recommendation by mining users' preference transition. CIKM '13, pages 733–738. ACM, 2013.

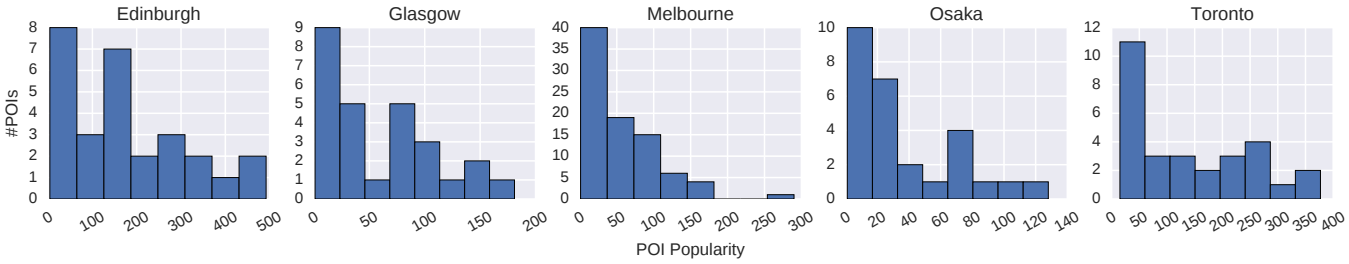
[27] Y. Liu, W. Wei, A. Sun, and C. Miao. Exploiting geographical neighborhood characteristics for location recommendation. CIKM '14, pages 739–748. ACM, 2014.

[28] 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.

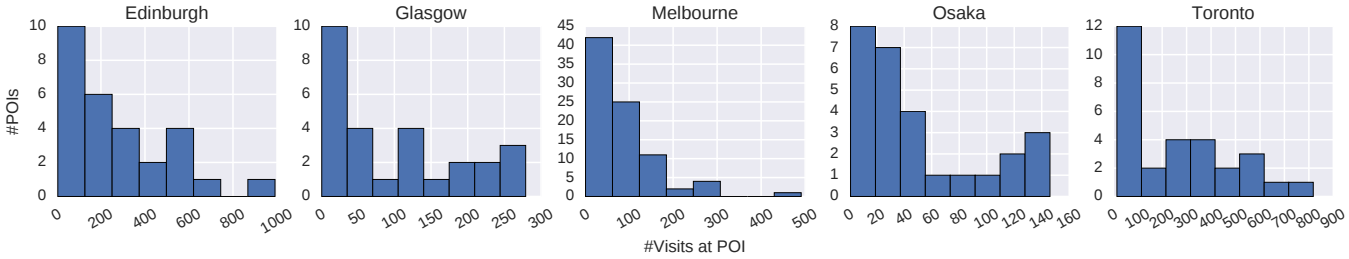
[29] X. Lu, C. Wang, J.-M. Yang, Y. Pang, and L. Zhang. Photo2Trip: Generating travel routes from geo-tagged photos for trip planning. MM '10, pages 143–152. ACM, 2010.

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

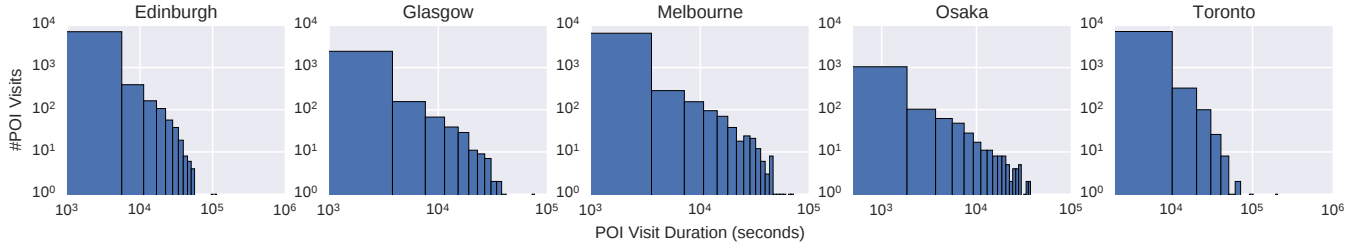
[31] D. Quercia, R. Schifanella, and L. M. Aiello. The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city. Hypertext '14, pages 116–125. ACM, 2014.



(a) Distribution of POI popularity



(b) Distribution of the number of visit at POI



(c) Distribution of the visit duration at POI

Figure 5: Distribution of POI popularity, the number of visit and visit duration

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

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

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

[35] 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.

[36] H. Yin, X. Zhou, Y. Shao, H. Wang, and S. Sadiq. Joint modeling of user check-in behaviors for point-of-interest recommendation. CIKM '15, pages 1631–1640. ACM, 2015.

[37] 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.

[38] 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.

[39] C. Zhang, H. Liang, K. Wang, and J. Sun. Personalized trip recommendation with POI availability and uncertain traveling time. CIKM '15, pages 911–920. ACM, 2015.

[40] J.-D. Zhang, C.-Y. Chow, and Y. Li. LORE: Exploiting sequential influence for location recommendations. SIGSPATIAL '14, pages 103–112. ACM, 2014.

[41] J.-D. Zhang, C.-Y. Chow, and Y. Zheng. ORec: An opinion-based point-of-interest recommendation framework. CIKM '15, pages 1641–1650. ACM, 2015.

[42] 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.

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

[44] 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.

[45] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. WWW '09, pages 791–800. ACM, 2009.

[46] 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.