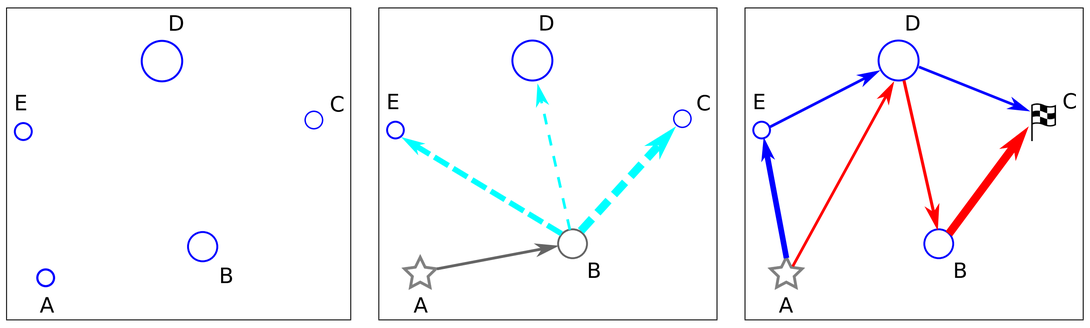
Learning Points and Routes to Recommend Trajectories

Dawei Chen, Cheng Soon Ong & Lexing Xie, Australian National University & Data61

1 The Problem



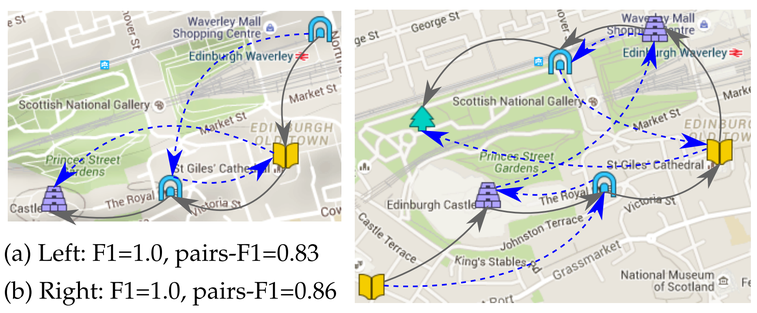
(a) POI Recommendation (b) Next Location Recommendation (c) Trajectory Recommendation

Figure 1: Three settings of trajectory recommendation problems. Instead of scoring POIs (Figure a) or recommending the next location (Figure b), we focus on recommending a whole trajectory (Figure c).

Problem formulation: given a set of POIs $\mathcal{P}$ and a *query* $q = (p_s, p_e, L)$, i.e., the start/end location and the number of desired POIs, we recommend a trajectory $\mathcal{T} = (p_s, \ldots, p_e)$ with L POIs.

2 Main Contributions

- Optimise point preferences and routes jointly.
- Feature-driven approach and learning from past behaviour.
- New evaluation metric that captures POI visiting orders.



(a) Left: F1=1.0, pairs-F1=0.83
(b) Right: F1=1.0, pairs-F1=0.86

Figure 2: Examples for F_1 vs pairs-F_1 as evaluation metric. pairs-F_1 lies between 0 and 1, and achieves 1 *iff* two trajectories are exactly the same.

To measure the similarity between two trajectories, we propose a new metric, pairs-F_1, that takes into account both the point identity and the visiting orders in a trajectory. It measures the F_1 score of every pair of ordered POIs: pairs-$F_1 = 2P_{\text{PAIR}}R_{\text{PAIR}} / (P_{\text{PAIR}} + R_{\text{PAIR}})$.

3 Methods

Rank POIs using POI and query specific features via rankSVM.

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

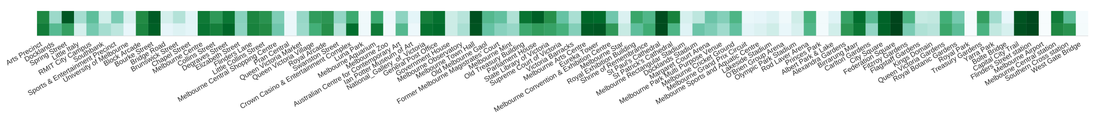


Figure 3: Examples of POI ranks in Melbourne. Top: POI ranks with respect to query (p_s = *Spring Street*, p_e = *Federation Square*, $L = 5$), the top-3 POIs are (*Flinders Street station, St Paul's Cathedral, Melbourne Town Hall*), the bottom-3 POIs are (*Royal Park, West Gate Bridge, Melbourne Airport*); Bottom: POI ranks with respect to query (p_s = *Federation Square*, p_e = *Melbourne Aquarium*, $L = 4$), the top-3 POIs are (*Flinders Street station, St Paul's Cathedral, Capital City Trail*), the bottom-3 POIs are (*West Gate Bridge, Royal Park, Melbourne Airport*).

Learn Routes

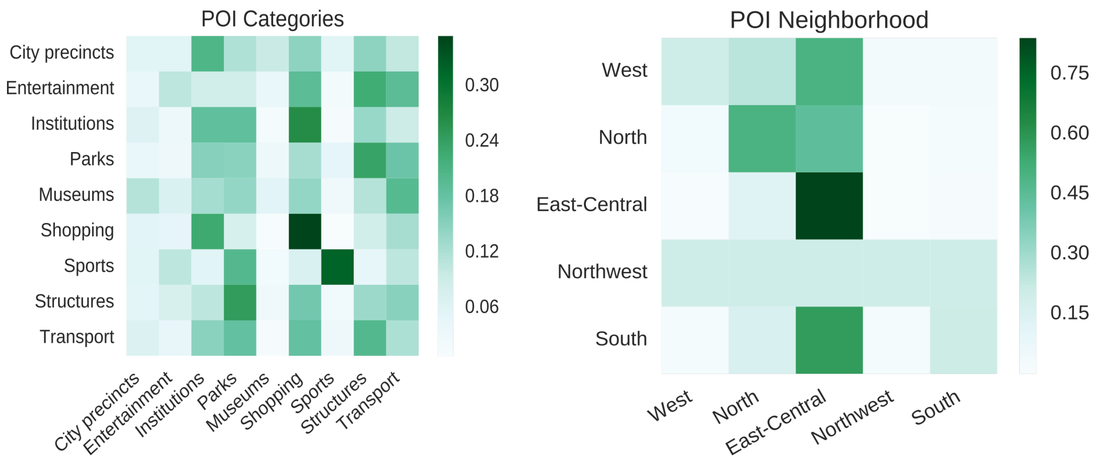


Figure 4: Model POI transitions using a Markov chain with discrete states and factorise the POI-POI transition matrix along different types of POI properties. Left: Transition matrix for POI category; Right: Transition matrix for POI neighborhood.

Combine Ranks and Routes

Maximise the ranking of POIs in trajectory as well as the transition likelihood simultaneously using Viterbi algorithm.

$$\underset{\mathcal{T} \in \mathcal{P}^L}{\text{argmax}}\ \alpha \sum_{k=2}^{L} \log P_R(p_k|q) + (1-\alpha) \sum_{k=1}^{L-1} \log P(p_{k+1}|p_k).$$

Avoid Sub-tours

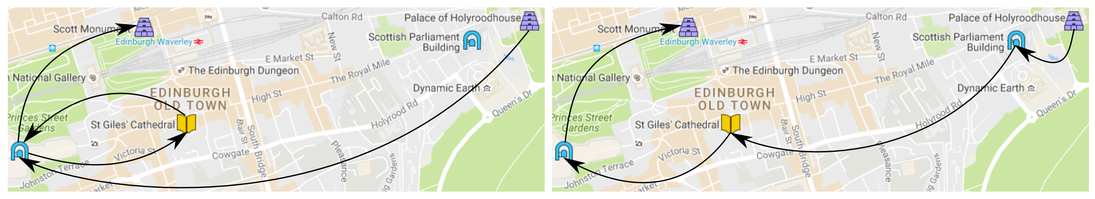


Figure 5: To avoid sub-tours in trajectories recommended by Viterbi algorithm, an integer linear program (ILP) with sub-tour elimination constraints was employed. Left: Trajectory recommended using Viterbi algorithm given a query; Right: Trajectory recommended by ILP with respect to the same query.

4 Experimental Results

	#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 1: Dataset of trajectories extracted from Flickr photos in five cities.

	Query	POI	Trans.	No sub-tours
RANDOM	×	×	×	×
PERSTOUR	×	√	×	√
PERSTOUR-L	×	√	×	√
POIPOPULARITY	×	√	×	×
POIRANK	√	√	×	×
MARKOV	×	√	√	×
MARKOVPATH	×	√	√	√
RANK+MARKOV	√	√	√	×
RANK+MARKOVPATH	√	√	√	√

Table 2: Information captured by nine different trajectory recommendation algorithms.

Performance Comparison

Table 3: Performance comparison on five datasets in terms of F_1 *score*. The best method for each city 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	0.656 ± 0.223	$\mathbf{0.801 \pm 0.213}$	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	$\mathbf{0.701 \pm 0.160}$	0.745 ± 0.166	0.620 ± 0.136	0.663 ± 0.125	0.678 ± 0.121
POIRANK	$\mathit{0.700 \pm 0.155}$	$\mathit{0.768 \pm 0.171}$	$\mathit{0.637 \pm 0.142}$	$\mathbf{0.745 \pm 0.173}$	$\mathbf{0.754 \pm 0.170}$
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	$\mathbf{0.639 \pm 0.146}$	$\mathit{0.732 \pm 0.162}$	$\mathit{0.751 \pm 0.170}$

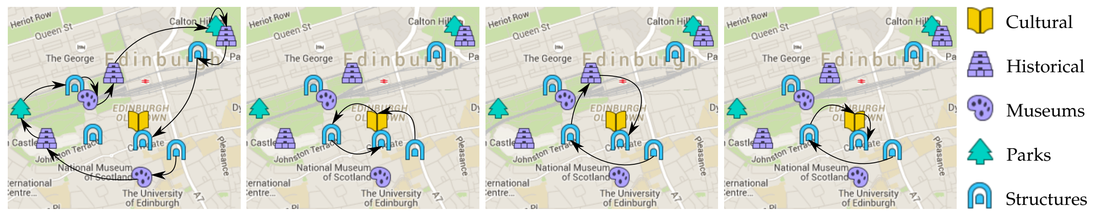
Table 4: Performance comparison on five datasets in terms of *pairs-F_1 score*. The best method for each city is shown in **bold**, the second best is shown in *italic*.

	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	0.417 ± 0.343	$\mathbf{0.643 \pm 0.366}$	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	$\mathit{0.436 \pm 0.259}$	0.507 ± 0.298	0.316 ± 0.178	0.365 ± 0.190	0.384 ± 0.201
POIRANK	0.432 ± 0.251	$\mathit{0.548 \pm 0.311}$	0.339 ± 0.203	$\mathbf{0.511 \pm 0.309}$	$\mathbf{0.518 \pm 0.296}$
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	$\mathbf{0.444 \pm 0.263}$	0.545 ± 0.306	$\mathbf{0.351 \pm 0.220}$	0.486 ± 0.288	0.512 ± 0.303
RANK+MARKOVPATH	0.428 ± 0.245	0.533 ± 0.303	$\mathit{0.344 \pm 0.206}$	$\mathit{0.489 \pm 0.287}$	$\mathit{0.514 \pm 0.297}$

Observations

- Learning-based approaches generally outperform heuristic route recommendation.
- Incorporating transitions to POI ranking results in a better sequence of POIs.
- Avoiding sub-tours improves performance of classical Markov chain methods.

An Illustrative Example



(a) PERSTOUR (b) POIRANK (c) MARKOV (d) Best

Figure 6: Example of different recommendations from algorithm variants. The best result (Figure d), with exactly the same points and routes as the ground truth, is achieved by RANK+MARKOVPATH.

Code/Data: `https://bitbucket.org/d-chen/tour-cikm16`