From Trajectories to Activities: A Spatio-Temporal Join Approach*

Kexin Xie, Ke Deng, Xiaofang Zhou

School of Information Technology & Electrical Engineering
University of Queensland
Brisbane, Australia
{kexin, dengke, zxf}@itee.uq.edu.au

* This work is partially supported by Microsoft Research Asia Internet Services Theme Research Program
Activity Sequences

- What do we do each day?
- E.g.
  - Drink after work?
  - Drunk after semester ends?
  - Jogging for 1 hour every Saturday?
  - Watching movie after 4 hours of shopping?
  - Only have one meal everyday during semester breaks?
  - Sleep when the sun rises?
Data Mining with Activity Sequences

- Technique => Knowledge
  - Frequent Sequential Pattern => Common Behavior (e.g. Collaborative promotions)
  - Periodic Pattern => Periodic Behavior (e.g. POI suggestion)
  - Outlier Detection => Odd Behavior (e.g. Criminal Investigation)
  - Clustering => Similar Behavior (e.g. Friend suggestion)
- The Problem?

Spend $80 or more at Gardencity, Get 20% Discount for movie tickets
Data Mining with Activity Sequences

- Technique => Knowledge
  - Frequent Sequential Pattern => Common Behavior (e.g. Collaborative promotions)
  - Periodic Pattern => Periodic Behavior (e.g. POI suggestion)
  - Outlier Detection => Odd Behavior (e.g. Criminal Investigation)
  - Clustering => Similar Behavior (e.g. Friend suggestion)

- The Problem?
  - Facebook vs Twitter

Spend $80 or more at Gardencity, Get 20% Discount for movie tickets
Data Mining with Activity Sequences

- Technique => Knowledge
  - Frequent Sequential Pattern => Common Behavior (e.g. Collaborative promotions)
  - Periodic Pattern => Periodic Behavior (e.g. POI suggestion)
  - Outlier Detection => Odd Behavior (e.g. Criminal Investigation)
  - Clustering => Similar Behavior (e.g. Friend suggestion)

- The Problem?
  - Facebook vs Twitter
  - How to obtain the activity sequences?
Travel Sequences

- Guess activity from travel sequences
  - From office to a bar every workday.
  - Stay at night club for 5 hours twice a year.
  - Moving slightly faster than working speed in the park every Saturday.
  - Moving slowly in shopping center for 4 hours then stays in a cinema.

- Where can we get such travel sequences?
  - GPS-enabled mobile devices
From Trajectory to Sequence of Activities

- **Problem**
  - Given a set of trajectories and a set of POIs, find the sequence of activities that might be performed during those set of trajectories

- **Rationale**
  - If the user stays at a POI for long enough time, then some activity may take place.

- **Question**
  - Which POIs did the user stay?
  - What activities the user performed?
An Influence Model

- Given a set of POIs $\mathcal{P}$, a segment of trajectory $T$ is *influenced* by a $p \in \mathcal{P}$ if for every point $pt$ on $T$, there does not exists a point $p' \in \mathcal{P}$, $\text{dist}(pt, p') < \text{dist}(pt, p)$.
- The *influence duration* of a POI $p$ given a segment of trajectory $T$ influenced by $p$, is the timestamp difference of $T$. 
Influence Example

Study@University ➞ Shopping@Supermarket ➞ Dining@Restaurant_B
Trajectories to Activities using Influence Duration

- How long is long enough?
  - Different POI have different requirement
  - e.g. Fast food restaurant vs Luxury restaurant

- Which activity took place?
  - Different activity requires different time length
  - e.g. Working in restaurant vs Dining in restaurant
A POI-Activity Mapping Set (PAMS) is a set of quadruple $M = \{(p, e, t_{\text{min}}, t_{\text{max}})\}$, where $p$ is some POI, $e$ is some activity, and $t_{\text{min}}$, $t_{\text{max}}$ are the minimum and maximum elapsed time for user activity $e$ happened at $p$. 

<table>
<thead>
<tr>
<th>POI</th>
<th>Activity</th>
<th>$t_{\text{min}}$</th>
<th>$t_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restaurant</td>
<td>Delivery</td>
<td>15 minutes</td>
<td>45 minutes</td>
</tr>
<tr>
<td>Restaurant</td>
<td>Dinning</td>
<td>30 minutes</td>
<td>2 hours</td>
</tr>
<tr>
<td>Restaurant</td>
<td>Working</td>
<td>3 hours</td>
<td>8 hours</td>
</tr>
<tr>
<td>University</td>
<td>Studying</td>
<td>2 hours</td>
<td>12 hours</td>
</tr>
<tr>
<td>University</td>
<td>Working</td>
<td>4 hours</td>
<td>12 hours</td>
</tr>
<tr>
<td>Petrol Shop</td>
<td>Fueling</td>
<td>10 minutes</td>
<td>30 minutes</td>
</tr>
<tr>
<td>Supermarket</td>
<td>Shopping</td>
<td>30 minutes</td>
<td>2 hours</td>
</tr>
<tr>
<td>Fast Food</td>
<td>Takeaway</td>
<td>10 minutes</td>
<td>45 minutes</td>
</tr>
<tr>
<td>Park</td>
<td>Playing</td>
<td>15 minutes</td>
<td>1 hour</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Assign User Activities

- Influence duration are checked against $t_{\text{min}}$ and $t_{\text{max}}$ in PAMS
- Multiple activities may happen.

E.g. Given a subtrajectory $T_s$ influenced by POI $p_i$ with influence duration 50, the PAMS is given as follow:

<table>
<thead>
<tr>
<th>POI</th>
<th>Event</th>
<th>$t_{\text{min}}$</th>
<th>$t_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_i$</td>
<td>a</td>
<td>15</td>
<td>45</td>
</tr>
<tr>
<td>$p_i$</td>
<td>b</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>$p_i$</td>
<td>c</td>
<td>40</td>
<td>90</td>
</tr>
</tbody>
</table>

The possible activity are:
- a and b
- b
- c
Trajectory Semantic Join

- Input: A set of trajectories $\mathcal{ST}$, a set of POIs $\mathcal{P}$ and a PAMS $\mathcal{M}$
- Output: All subtrajectory-activity doubles $(T_s, e)$, such that $e$ may happen at $T_s$

$$\exists_{T \in \mathcal{ST}}. T_s \in T, \text{ and } \exists_p \in \mathcal{P}. E = EA(p, idr(p, T_s), M)$$

((T_1, (\text{Studying@University OR Working@University})), (T_2, \text{Shopping@Supermarket}), (T_3, \text{Dining@Restaurant}))
Filter and Refine Approach

- **Filter**: *Trajectory-POI join* - find all subtrajectory-POI pairs that activities may took place

- **Refine**: assign possible events to these subtrajectory-POI pairs
Trajectory Semantic Join using Voronoi Diagram

- Voronoi Diagram
  - A Voronoi diagram of a set of points $P$ is the subdivision of the plain into $n$ cells, one of each site in $P$, with the property that a point $q$ lies in the cell corresponding to a site $p_i$ if and only if $\text{dist}(p, p_i) < \text{dist}(q, p_j)$ for each $p_j \in P$, with $j \neq i$.

- Computes Voronoi Diagram
  - Fortune’s Algorithm to Compute Voronoi diagram in $O(n \log n)$ time.
Trajectory Semantic Join using Voronoi Diagram

- Voronoi Diagram
  - A Voronoi diagram of a set of points $P$ is the subdivision of the plain into $n$ cells, one of each site in $P$, with the property that a point $q$ lies in the cell corresponding to a site $p_i$ if and only if $\text{dist}(p, p_i) < \text{dist}(q, p_j)$ for each $p_j \in P$, with $j \neq i$.

- Computes Voronoi Diagram
  - Fortune’s Algorithm to Compute Voronoi diagram in $O(n \log n)$ time.
Algorithm 1: Trajectory_POI_Join

Input: A set of trajectories \( ST \), a set of POIs \( P \), and its Voronoi diagram \( V \).
Output: A set of triples (subtrajectory, POI, actual influence duration)

/* Step 1 - Compute line intersections */
1. Compute line intersection points of all trajectories in \( ST \) and all edges in \( V \) using plain-sweep technique;

/* Step 2 - Organize and Output */
2. Group the intersection points by trajectory and sort them by their timestamp;
3. \( \text{foreach} \ T \in ST \) do
4. \( \text{foreach} \ \text{consecutive intersection points} \ p_i \ \text{and} \ p_j \ \text{in} \ T \) do
5. Find the subtrajectory \( T_s \) of \( T \) defined enclosed by \( p_i \) and \( p_j \);
6. Find common POI \( p \) that influence both \( p_i \) and \( p_j \);
7. Find the time difference \( d \) of \( p_i \) and \( p_j \);
8. Output the triple \( (T_s, p, d) \);
TP Join (Compute Line Intersections)

- Plane-sweep for Line Intersections
  - Line segment becomes active when intersect with the sweep line
  - Only “active” line segments can intersect with each other
  - Perform intersection check between “active” line segments
Observation: Given a trajectory intersects Voronoi edges on a sequence of points \((p_0, p_1, ..., p_n)\), where \(\forall 0 \leq i < n \ timestamp(p_i, T) \leq timestamp(p_{i+1}, T)\). All consecutive points \(p_i\) and \(p_{i+1}\) \((0 \leq i < n)\) have at least one common influenced POI influencing POI \(p_c\), and the subtrajectory \(T_i\) of \(T\), defined by \(p_i\) and \(p_{i+1}\) is influenced by \(p_c\).

Temporally sorted intersection points enclosed subtrajectories are fully influenced by some POI.
Partition Based Trajectory Semantic Join (PTSJ)

- Problem with TP Join
  - Memory based techniques
  - Need sort Voronoi edges and segment of trajectories globally

- Partition Based TSVID Join (PTSJ)
  - A modified algorithm from PBSM (Patel SIGMOD 96)
  - Partition the space into smaller size grids that spatial object can be fit into memory, insert the Key-Pointer-Element (KPE) of spatial objects into each partition if their MBR overlap with the partition.
  - Estimate number of partitions:
    \[
    pp = \left\lfloor \frac{(|T| + |V|) \times \text{sizeof}(KPE)}{M} \right\rfloor
    \]
  - Perform in memory technique for each partition.
  - Repartition may be needed if not enough memory.
Partition Based Trajectory Semantic Join (PTSJ)

Algorithm 2: PTSJ

Input: A set of trajectories $ST$, a set of POIs $P$ with their influence regions $VC$, and a PAMS $L$
Output: A set of TPT and event pairs

1. $p \leftarrow$ Estimate number of partitions, if not specified in the input;
2. Partition table $G \leftarrow$ Partition the space into $p$ partitions;
3. foreach $T \in ST$ do
   4. $G' \leftarrow$ The set of partitions that $T.mbr$ overlap with;
   5. Insert KPE of $T$ into partitions in $G'$;
4. foreach $vc \in VC$ do
   5. $G' \leftarrow$ The set of partitions that $V.mbr$ overlap with;
   6. Insert KPE of $vc$ into partitions in $G'$;
9. foreach partition $g \in G$ do
   10. Load All Voronoi cells $VC'$ and Trajectories $ST'$ in $g$ into internal memory;
   11. $TPTS \leftarrow$ Trajectory.POIJn ST, VC';
   12. Assign events for each TPT in TPTS and output the result;
Problems with PTSJ

- Trajectories and Voronoi cells may be inserted into multiple partition entries.
- Many duplicated entries
Reuse duplicated entries

- Duplication Reuse

  - Each KPE stores the partitions the spatial object is inserted in, when retrieve the spatial objects of the next partition, the spatial objects with duplicated KPE do not need to be loaded in to memory (save I/O cost).

  - Order of performing computations for the partitions are important.
Number of duplicated entries

- If $\text{KPE}_o$ is duplicated with partition $P$ and one of its opposite partitions $P_{\text{opp}}$, then it must have duplicated entries in one of its adjacent partitions.

- If $\text{KPE}_o$ is duplicated with partition $P$ and one of its disjoint partitions $P_{\text{dist}}$, then it must have duplicated entries in one of its adjacent partitions.

- Hence $N_{\text{dup}}(P, \text{adj}(P)) \geq N_{\text{dup}}(P, \text{opp}(P)) \geq N_{\text{dup}}(P, \text{disj}(P))$

- Hillbert order always traverse adjacent partitions one after the other.
PTSJ+ - PTSJ with duplication reuse

Algorithm 3: PTSJ+

Input: A set of trajectories \( ST \), a set of POIs \( P \) with their influence regions \( VC \), and a PAMS \( L \)

Output: A set of TPT and event pairs

1. \( p \leftarrow \) Estimate number of partitions if not specified in input;
2. Partition table \( T \leftarrow \) Partition the space into \( p \) partitions;
3. foreach \( T \in ST \) do
   4. \( G' \leftarrow \) The set of partitions that \( T.mbr \) overlap with;
   5. Insert KPE of \( T \) into partitions in \( G' \);
   6. Annotate each KPE with all the partitions they are inserted to;
4. foreach \( vc \in VC \) do
   7. \( G' \leftarrow \) The set of partitions that \( V.mbr \) overlap with;
   8. Insert KPE of \( vc \) into partitions in \( G' \);
   9. Annotate each KPE with all the partitions they are inserted to;
10. Sort the partitions in \( G \) using Hillbert order;
11. Shared Voronoi cells \( SSV \leftarrow \emptyset \);
12. Shared Trajectories \( SST \leftarrow \emptyset \);
13. foreach partition \( g \in G \) do
14. \( VC' \leftarrow SSV \cup \) load Voronoi Cells in \( g \) but not in \( SSV \);
15. \( SSV \leftarrow \) Voronoi cells in both \( g.Content\)s and \( g.next.Content\)s;
16. \( ST' \leftarrow \emptyset \);
17. foreach KPE \( k \in g.TrajectoryContents \) do
18. if \( k.id \in SST.ids \) then
19. \( ST' \leftarrow ST' \cup \{SST.get(k.id)\} \);
20. else
21. \( ST' \leftarrow ST' \cup \{\text{loadFromDisk}(k.id)\} \);
22. \( TPTS \leftarrow \text{Trajectory\_POI\_Join}(ST', VC') \);
23. Assign events for each TPT in \( TPTS \) and output result;

- Insert KPEs of each spatial object into each partition.
- Each KPEs include the IDs of all partitions it is inserted into.
- Partitions need to be sorted in Hillbert order.

- Do TP-Join in for each partition in Hillbert order.
- Keep track of shared spatial objects
- Avoid re-loading already loaded spatial objects
Experiments

- Experimental Setting
  - POI: 6487 POIs cropped from 105725 California POIs
  - Trajectories: simulated from California road networks

<table>
<thead>
<tr>
<th></th>
<th>Drive</th>
<th>Walk</th>
<th>Stop_s</th>
<th>Stop_m</th>
<th>Stop_l</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average Speed</td>
<td>50</td>
<td>5</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(Km/Hour)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Switching Mode</td>
<td>$\frac{1}{50}$</td>
<td>$\frac{1}{15}$</td>
<td>$\frac{1}{20}$</td>
<td>$\frac{1}{120}$</td>
<td>$\frac{1}{240}$</td>
</tr>
<tr>
<td>(probability)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Can Switch To</td>
<td>Walk</td>
<td>All</td>
<td>Walk</td>
<td>Walk</td>
<td>Walk</td>
</tr>
<tr>
<td>(Mode)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Time consumed by building index, retrieving spatial objects (I/O) is measured.
Experimental Result

- Elapsed Time (sec)
  - Number of Trajectories
    - PTSJ
    - PTSJ+

- Trajectory Length (Hours)
  - PTSJ
  - PTSJ+
Conclusion

- Trajectory and Semantic Issues:
  - Preprocessing for Data Mining
  - Understand User Behaviours
- Future works
  - Further Optimisations
  - Privacy Issues
Questions?