PublicTransitSnapper:

Dynamic Map-Matching To Public Transit Vehicles

Bachelor’s Thesis by Robin Wu
Introduction
Problem Definition

$n$ timestamped GPS Points

\[ g = [pt_1, \ldots, pt_n] \]

Most likely public transit vehicle
Approach
GTFS Dataset

General Transit Feed Specification (GTFS)

Each trip is described by:
- Shape
- Route
- Service
  - Active weekdays
  - Exception dates
- Stops
  - Location
  - Stop times

PublicTransitSnapper: Dynamic Map-Matching To Public Transit Vehicles
Close Edges
Active Edges

- Check service for the given date

- Time information available for stops

- Check $time_{start} - \varepsilon \leq t \leq time_{end} + \delta$
Hidden Markov Model (HMM)
**HMM Cost Functions**

- Convert to log-space → summation

\[ C_{\text{emission}}(e|pt) = ||e, pt|| \]

\[ C_{\text{transition}}(e_1 \rightarrow e_2) = ||e_1|| + ||e_2|| + ||\text{shortest\_path}(\text{end}(e_1), \text{start}(e_2))|| + \text{direction\_penalty}(e_1, e_2) \]
Most Likely Shape
Adding Time Data

- Multiple trips can be active at the same time
Algorithm

- Find all close edges
- Filter the close edges for active edges
- Determine the most likely shape with a HMM
- Select the most likely trip from the most likely shape
Evaluation
**Evaluation Method**

- Generate own test dataset with GTFS dataset
  - Generate test data for each trip with multiple tests
  - User travels on a trip for 4 stops with 10 GPS Points
  - Generate a timestamp for each GPS Point

- Add noise
  - GPS inaccuracy $\mathcal{N} \sim (0, 16)$
  - Stop times $\max(0, \mathcal{N} \sim (0, 60))$
  - Timestamps $\mathcal{N} \sim (0, 30)$
Accuracy Measure

\[ 1(test) = \begin{cases} 1, & \text{if matched correct trip} \\ 0, & \text{otherwise} \end{cases} \]

- Number of tests depends on number of stops

\[
\text{accuracy} = \frac{1}{|\text{Test Data}|} \sum_{td \in \text{Test Data}} \left( \frac{1}{|td|} \sum_{test \in td} 1(test) \right)
\]
## Evaluation Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>total trips</th>
<th>tram</th>
<th>bus</th>
<th>funicular</th>
<th>train</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freiburg</td>
<td>19,153</td>
<td>9,063</td>
<td>10,090</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SWEG</td>
<td>733</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>733</td>
</tr>
<tr>
<td>Zürich</td>
<td>33,178</td>
<td>—</td>
<td>31,971</td>
<td>1,206</td>
<td>—</td>
</tr>
</tbody>
</table>
## Baseline Algorithms

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average Accuracy</th>
<th>Average Run-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
<td>BaselineHMM</td>
</tr>
<tr>
<td>Freiburg</td>
<td>0.4%</td>
<td>0.8%</td>
</tr>
<tr>
<td>SWEG</td>
<td>0.8%</td>
<td>1.6%</td>
</tr>
<tr>
<td>Zürich</td>
<td>0.2%</td>
<td>0.4%</td>
</tr>
</tbody>
</table>
## Evaluation Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average Accuracy</th>
<th>Average Run-Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ActiveEdges</td>
<td>TimeAfter</td>
</tr>
<tr>
<td>Freiburg</td>
<td>91.2%</td>
<td>91.3%</td>
</tr>
<tr>
<td>SWEG</td>
<td>32.5%</td>
<td>32.3%</td>
</tr>
<tr>
<td>Zürich</td>
<td>92.3%</td>
<td>94.8%</td>
</tr>
</tbody>
</table>

with allowed time “slack” $\varepsilon = 1, \delta = 5 \text{ min}$
SWEG Dataset

- Trains have higher distances between stops

Reducing the number of stops in the test data for the SWEG dataset
Live Demo
Real-Time Data

- GTFS Realtime

- Update for a trip:
  
  ```json
  ["stop_sequence": 1, "departure": {"delay": 5}
  "stop_sequence": 3, "departure": {"delay": 10}]
  ```

- Apply the delay to the stop times

  15:00:00 + 5s
  15:05:00 + 5s
  15:10:00 +10s
R-Tree

- A graph does not support efficient spatial queries
- Insert edges into an R-Tree
- Lookup on average $O(\log(n))$
“Overtime” Trips

- Trips can be in “overtime”
  - a trip runs on Monday from 23:30:00 till 25:30:00 (Tuesday 01:30:00)
    → active weekdays only contains Monday

- Checking active weekdays from the service can fail
  - User on Tuesday 01:00:00

- Generate \{(0,23,\text{False}),(1,0,\text{True}),(1,1,\text{True})\}

- For the user check (1,0,\text{False}) and (1,0,\text{True})
Determine the Next Stop

(a) $d_{s,e} > d_{p,e}$

(b) $d_{s,e} < d_{p,e}$
## Storage and Memory Consumption

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GTFS size</th>
<th>precompute size</th>
<th>precompute time</th>
<th>docker memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freiburg</td>
<td>28.3MB</td>
<td>36.1MB</td>
<td>1.65s</td>
<td>384MiB</td>
</tr>
<tr>
<td>SWEG</td>
<td>4.6MB</td>
<td>5.6MB</td>
<td>0.27s</td>
<td>148MiB</td>
</tr>
<tr>
<td>Zürich</td>
<td>45.5MB</td>
<td>58.0MB</td>
<td>2.47s</td>
<td>627MiB</td>
</tr>
</tbody>
</table>
“Broken” Trips

Two partly indistinguishable trips in the Freiburg dataset