Extraction of public transportation networks from Openstreetmap data and estimation of OD patterns through a graph convolutional approach

Koji Fukuda, Saeed Maadi, Kai Shen and Jan-Dirk Schmöcker

TransitData2019, 10 July 2019, Paris
Introduction

• Growing number of publically available “TransitData” and other transportation network data.
  
  • GTFS data.
  
  • Live feeds of bus positional data.
  
  • Openstreetdata.
  
  • Not transit: Published “complete data” (OD demand, network, flows) from cities for research purposes.
Motivation

- Utilisation of “Openstreetdata” for learning relation between urban structure and network flows.
  - Considering the cyclic relationship between urban form and network flows, how well does urban form explain network flows?
  - Specifically, are (public transport) OD patterns explained by infrastructure (and what kind of infrastructure)? Or how much is intrinsic demand?
  - Can reduced car traffic flows between OD pairs be explained by the existence of public transport?
Literature Classification

• OD estimation fundamental for planning and well established. Broad classification:

  • Gravity model type approaches with production, attraction and deterrence calibration (…)
    • Usually limited to a (fairly small) set of predetermined factors hypothesised to determine demand

  • Based on traffic or PT flow data (see Hickman (2018) for a review on OD estimation with smart card data)

  • Recent efforts to directly utilize network characteristics and topology as demand predictors (Luo et al, 2018 as paper on PT networks)

  • Deep learning approaches as tool to combine these approaches (Xiong et al, 2019)
OpenStreetMap

- OpenStreetMap is a free, editable map of the whole world that is being built by volunteers with an open-content license.
- Including not only road network but also public transportation (bus/train...)
- Various types of "Point of Interests" (POIs) (house, building, shop, school, station, park,...)

https://www.openstreetmap.org/
OpenStreetMapX.jl

OpenStreetMapX.jl (https://github.com/pszufe/OpenStreetMapX.jl)

OpenStreetMapX.jl

This package provides basic functionality for parsing, viewing, and working with OpenStreetMap map data. The package is intended mainly for researchers who want to incorporate this rich, global data into their work, and has been designed with both speed and simplicity in mind, especially for those who might be new to Julia.

Note: Our automated tests currently fail on Julia 0.4, but these problems appear to be contained to the Travis test system and are caused by a dependent package. OpenStreetMap.jl should run without issue on both Julia 0.3 and 0.4, and the tests pass on local machines for Julia 0.3.7 and 0.4.2 on OS X 10.11 and Ubuntu 14.04.

Capabilities

- Parse an OpenStreetMap XML datafile (OSM files)
- Crop maps to specified boundaries
- Convert maps between LLA, ECEF, and ENU coordinates
- Extract highways, buildings, and tagged features from OSM data
- Filter data by various classes:
  - Ways suitable for driving, walking, or cycling
  - Freeways, major city streets, residential streets, etc.
  - Accommodations, shops, industry, etc.
- Draw detailed maps using Julia's Winston graphics package with a variety of options
- Compute shortest or fastest driving, cycling, and walking routes using Julia's Graphs package
To facilitate “automatic” user-defined extraction of network features we create a software with 6 modules.

- **Road.m**: specify and extract links & nodes
  - s-link
  - intersection
  - waypoints
- **POI.m**: specify and extract OSM POIs
  - POIs
- **Bus.m**: extract links & nodes
  - b-link
  - b-node
- **Rail.m**: extract links & nodes
  - r-link
  - r-node
- **Manual POIs**:
- **Convert.m**:
  - x-link
  - x-node
- **Combine.m**: Associate POIs, bnode, rnode and MPOIs with usually nearest intersection
  - c-link.m
  - c-node.m
- **xgraph.gml**:
Resulting graph structure from software

- Graph structure
  - Node: Points of Interest (POI), Road intersections
    attributes: Type (station, park, shop, ...)
  - Link: road, railways,
    attributes: road width, # of lanes, sidewalk, name of railway,...
Public Transport network extraction: Bus

- Bus routes represented by a ‘route’ relation, includes bus stops and edges
- Often several stands associated with a single stop
- Stand association with line in some cases not clear
- To simplify network representation nearby stands combined into a single stop.
Public Transport network extraction: Bus

- Bus routes represented by a ‘route’ relation, includes bus stops and edges
- Often several stands associated with a single stop
- Stand association with line in some cases not clear
- To simplify network representation nearby stands combined into a single stop.

- OSM includes bus stops without bus line association;
- Our approximation: associate bus lines within 50m with the stop
Area to point conversion required and similar issues as for bus:

Merge stop nodes to one station.
If stop-node association with station-node is unknown, then associate with the nearest station-node.
Similar issues as for bus:

Merge stop nodes to one station.
If stop-node association with station-node is unknown, then associate with the nearest station-node.
Public Transport network extraction: Rail

Similar issues as for bus:

Merge stop nodes to one station. If stop-node association with station-node is unknown, then associate with the nearest station-node.
Supervised machine learning using (Deep) Graph Convolutional Network: In contrast to other NN methods, GCN aim to learn directly considering (random) graph structure and set of node features.

① extract road and PT network from OpenStreetMap
② extract POIs House, Shop, School, Park, Station, etc….
③ Combine POIs and road network
④ Learn the relationship between land use and OD flow.

OpenStreetMap data
Map+POIs
OD flow data (supervised learning)
Graph Convolutional Network (GCN)

- Deep neural network on graph structure.
- Each node gathers and summarizes the value of the neighboring nodes.
- By stacking multiple layers, the information of distant nodes can be considered.

\[ h_i^{(l+1)} = \sigma \left( W_0^{(l)} h_i^{(l)} + \sum_{j \in N_i^r} W^{(l)} h_j^{(l)} \right) \]

values on \((l + 1)\)-th layer
values on \(l\)-th layer

nonlinear activation (ReLU function)
self loop
sum over neighboring nodes
Transportation Networks (TNTP) dataset

https://github.com/bstabler/TransportationNetworks

- OD flow data mainly for studying the traffic assignment problem.

<table>
<thead>
<tr>
<th>Network</th>
<th>Zones</th>
<th>Links</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anaheim</td>
<td>38</td>
<td>914</td>
<td>416</td>
</tr>
<tr>
<td>Austin</td>
<td>7388</td>
<td>18961</td>
<td>7388</td>
</tr>
<tr>
<td>Barcelona</td>
<td>110</td>
<td>2522</td>
<td>1020</td>
</tr>
<tr>
<td>Berlin-Center</td>
<td>865</td>
<td>28376</td>
<td>12981</td>
</tr>
<tr>
<td>Berlin-Friedrichshain</td>
<td>23</td>
<td>523</td>
<td>224</td>
</tr>
<tr>
<td>Berlin-Mitte-Center</td>
<td>36</td>
<td>871</td>
<td>398</td>
</tr>
<tr>
<td><strong>Berlin-Mitte-Prenzlauerberg-Friedrichshain-Center</strong></td>
<td>98</td>
<td>2184</td>
<td>975</td>
</tr>
<tr>
<td>Berlin-Prenzlauerberg-Center</td>
<td>38</td>
<td>749</td>
<td>352</td>
</tr>
<tr>
<td>Berlin-Tiergarten</td>
<td>26</td>
<td>766</td>
<td>361</td>
</tr>
<tr>
<td>…</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Grey points: POIs (total number = 40,582)
- Red line: Road network from TNTP data
- Blue line: Road network extracted from OpenStreetMap
- Green line: Bus network extracted from OpenStreetMap
<table>
<thead>
<tr>
<th>POI types</th>
<th>number</th>
<th>POI types</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>building_yes</td>
<td>12,916</td>
<td>amenity_post_box</td>
<td>331</td>
</tr>
<tr>
<td>building_residential</td>
<td>6,056</td>
<td>building_garage</td>
<td>331</td>
</tr>
<tr>
<td>building_apartments</td>
<td>3,511</td>
<td>shop_hairdresser</td>
<td>315</td>
</tr>
<tr>
<td>amenity_bench</td>
<td>2,252</td>
<td>amenity_kindergarten</td>
<td>313</td>
</tr>
<tr>
<td>amenity_restaurant</td>
<td>1,365</td>
<td>amenity_atm</td>
<td>305</td>
</tr>
<tr>
<td>landuse_grass</td>
<td>828</td>
<td>amenity_bar</td>
<td>304</td>
</tr>
<tr>
<td>amenity_bicycle_parking</td>
<td>748</td>
<td>natural_scrub</td>
<td>289</td>
</tr>
<tr>
<td>amenity_cafe</td>
<td>747</td>
<td>building_school</td>
<td>288</td>
</tr>
<tr>
<td>amenity_waste_basket</td>
<td>730</td>
<td>amenity_pub</td>
<td>287</td>
</tr>
<tr>
<td>building_commercial</td>
<td>564</td>
<td>shop_convenience</td>
<td>287</td>
</tr>
<tr>
<td>shop_clothes</td>
<td>558</td>
<td>historic_memorial</td>
<td>279</td>
</tr>
<tr>
<td>amenity_fast_food</td>
<td>556</td>
<td>shop</td>
<td>251</td>
</tr>
<tr>
<td>leisure_playground</td>
<td>509</td>
<td>tourism_artwork</td>
<td>248</td>
</tr>
<tr>
<td>amenity_vending_machine</td>
<td>463</td>
<td>office_company</td>
<td>243</td>
</tr>
<tr>
<td>building_house</td>
<td>424</td>
<td>shop_supermarket</td>
<td>223</td>
</tr>
<tr>
<td>parking_surface</td>
<td>422</td>
<td>sport_table_tennis</td>
<td>213</td>
</tr>
<tr>
<td>amenity_recycling</td>
<td>388</td>
<td>amenity_parking</td>
<td>212</td>
</tr>
<tr>
<td>shop_bakery</td>
<td>348</td>
<td>tourism_information</td>
<td>204</td>
</tr>
<tr>
<td>amenity_telephone</td>
<td>346</td>
<td>landuse_village_green</td>
<td>186</td>
</tr>
<tr>
<td>building_roof</td>
<td>337</td>
<td>building_terrace</td>
<td>181</td>
</tr>
</tbody>
</table>
**Model Architecture**

- Generalized gravity model
- GCN architecture
  - Number of Layers = 2, 3, 4
  - Number of neurons in hidden layers = 16, 32, 48, 64

```
Feature vector at the Origin (251-dim)
```
```
Feature vector at the Destination (251-dim)
```
```
GCN
```
```
GCN
```
```
Path feature from O to D (2 or 4-dim)
- Distance
- Free Flow time
- Existence of bus route (0 or 1)
- Num. of stops on the bus route

*Estimated OD flow (scalar)*

```
5-layer Multi-Layer Perceptron (MLP)
```
```
dim = 2/4
```
```
dim = 16/32/48/64
```
```
dim = 16/32/48/64
```
```
Feature vector from the Origin (251-dim)
```
```
Feature vector at the Destination (251-dim)
```

*Each element of the vector represents the number of POIs of the type.*
OD data

Number of OD points (zone centroids): 98
Number of observed OD pairs: 9501

For training: 7604 ODs

For testing: 1901 ODs
- Test losses with changing the number of the layers and the neurons
- About 5% better by using bus route information
  → Existence of bus route affects the demand of car traffic.
- 4-layer, Number of neurons = 64, with bus route information
- Loss: mean squared error (MSE) between genuine and estimated OD values
Estimation result (scatter plot)

For test: 1901 ODs

Estimated value vs. True value
Conclusions and next steps

• Public transport data within OpenStreetData as “TransitData”?

• Tool to extract networks developed: Allows for easy comparison of different networks.

• Range of possible applications related to network design questions: (mis-)match of urban form and service provision.

• Application presented here is OD estimation
  • GCN as promising approach, improvement of estimates through inclusion of bus routes observed.

• Further work
  • Applying and comparison of networks; learning from one network apply to another network
  • Application to Smartcard data