**Data Temperature Formula**

**Martijn van Exel for TeleNav**

# Revision history

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Comments |
| 1 | 12/6/2011 | MvE | Initial version |

# Core formula

datatemp = RELATION\_WEIGHT \* self.relation\_temperature() + ROUTING\_WEIGHT \* self.routing\_attributes\_temperature() + FRESHNESS\_WEIGHT \* (cost\_ages1 + cost\_ages10 + cost\_ages25 + cost\_user95 + cost\_ages50 + cost\_ages75) \* BASIC\_TEMP + TIGER\_WEIGHT \* self.ways\_entity.RCM.tiger\_cost() \* BASIC\_TEMP + ZERO\_DATA\_TEMPERATURE

# Components

## Relation temperature

RELATION\_WEIGHT \* self.relation\_temperature()

RELATION\_WEIGHT is a constant set to 0.1.

self.relation\_temperature() returns the sum of the length of all relations that represent a turn restriction, divided by the total length of all relations.

### Suggestions

#### Normalization

Turn restrictions typically represent very simple relations: two way segments and the node that represents their intersection. It does not seem right to normalize on relation length for two reasons:

1. The turn restriction relation says something about the ways that are in the turn restriction, not about the quality of all relation features.
2. The length is not a proper normalization entity. Relations can be very long (think administrative and even country borders) and turn restrictions are typically really short because they only consist of two way segments.

A better – but still far from perfect – way to represent turn restriction presence would be to relate the number of turn restrictions to the length of ways that are not tagged with oneway=1/-1/true, and are of a navigable road type where turn restrictions are not implicit (primary, secondary, tertiary, residential, unclassified).

#### More relations

Currently, we’re only looking at turn restriction relations. This is the only relation type that is directly relevant to routing quality, but the presence of other relation types like bridge, tunnel, route does indicate that there are advanced mappers active in the area, which would generally reflect positively on data quality. See hard versus soft qualityXXX

## Routing temperature

ROUTING\_WEIGHT \* self.routing\_attributes\_temperature()

RELATION\_WEIGHT is a constant set to 0.1.

self.routing\_attributes\_temperature() returns the weighed result of cost functions for binned way features. This is a more complex function, so let’s break it down.

Way feature bins – all OSM way features are binned into a number of classes pertaining to routing relevance:

* Guidance
* Highway
* Local
* Main
* Unclassified

This is the complete class map from OSM way classes to ransm classes:

|  |  |
| --- | --- |
| osm class | ransm class |
| cycleway | guidance |
| footway | guidance |
| mini\_roundabout | guidance |
| stop | guidance |
| give\_way | guidance |
| traffic\_signals | guidance |
| crossing | guidance |
| roundabout | guidance |
| motorway\_junction | guidance |
| turning\_circle | guidance |
| construction | guidance |
| motorway | highway |
| secondary | local |
| tertiary | local |
| residential | local |
| living\_street | local |
| service | local |
| track | local |
| pedestrian | local |
| raceway | local |
| services | local |
| rest\_area | local |
| bus\_guideway | local |
| path | local |
| motorway\_link | local |
| trunk\_link | local |
| primary\_link | local |
| secondary\_link | local |
| tertiary\_link | Local |
| trunk | Main |
| primary | Main |
| unclassified | unclassified |
| road | unclassified |

For each of the bins, as well as for the remainder bin (called uncommon), a cost function is used to calculate a partial routing temperature. The cost function is defined in attribute\_cost(self, road\_category) in entity.py. The basic function is

length\_cost \* LENGTH\_COST + routing\_cost \* ROUTING\_COST + junction\_cost \* JUNCTION\_COST + tiger\_cost \* TIGER\_COST

and thus breaks down in four dimensions:

1. **Length** – the total length of ways in this bin divided by the total length of all ways.
2. **Routing** – further broken down in three weighed factors
   1. Oneway (0.45) – The total length of ways having a oneway tag divided by the total length of ways in the bin
   2. Maxspeed (0.45) – The total length of ways having a maxspeed tag divided by the total length of ways in the bin
   3. Access (0.1) – The total length of ways having an access tag divided by the total length of ways in the bin
3. **Junction** – the number of features having a junction tag divided by the total number of features in that bin
4. **TIGER** – further broken down in two weighed factors:
   1. **Untouched by users (-0.3)** – the number of TIGER ways that are still at version 1, versus the total number of ways that have TIGER tags.
   2. **Version increase over TIGER (**0.7) – the sum of version increase count over TIGER divided by the sum of all TIGER tagged way versions

The partial temperatures are then weighed according to this weight map:

ROAD\_CATEGORY\_WEIGHTS = {'highways': 0.3, 'main': 0.20, 'local': 0.10, 'guidance': 0.2, 'unclassified':-0.1, 'uncommon':-0.1}

|  |  |
| --- | --- |
| Bin | Relative weight |
| Highways | 0.3 |
| Main | 0.2 |
| Local | 0.1 |
| Guidance | 0.2 |
| Unclassified | -0.1 |
| Uncommon | -0.1 |

### Suggestions

Unclassified is not negative.

Tiger cost does not add up to 1 (-0.3 + 0.7 = 0.4)

## Freshness temperature

FRESHNESS\_WEIGHT \* (cost\_ages1 + cost\_ages10 + cost\_ages25 + cost\_user95 + cost\_ages50 + cost\_ages75) \* BASIC\_TEMP

### Suggestions

## Tiger temperature

TIGER\_WEIGHT \* self.ways\_entity.RCM.tiger\_cost() \* BASIC\_TEMP

### Suggestions

## Baseline

ZERO\_DATA\_TEMPERATURE