**Matthew R. Versaggi**

Artificial Intelligence Engineer - Imagine One Technology and Management, Ltd.

Greater Minneapolis-St. Paul Area | Defense & Space

Current Imagine One Technology and Management, Ltd.

Previous Versaggi Information Systems Inc., DePaul University, (EBS) Electronic Business Systems

Education Udacity

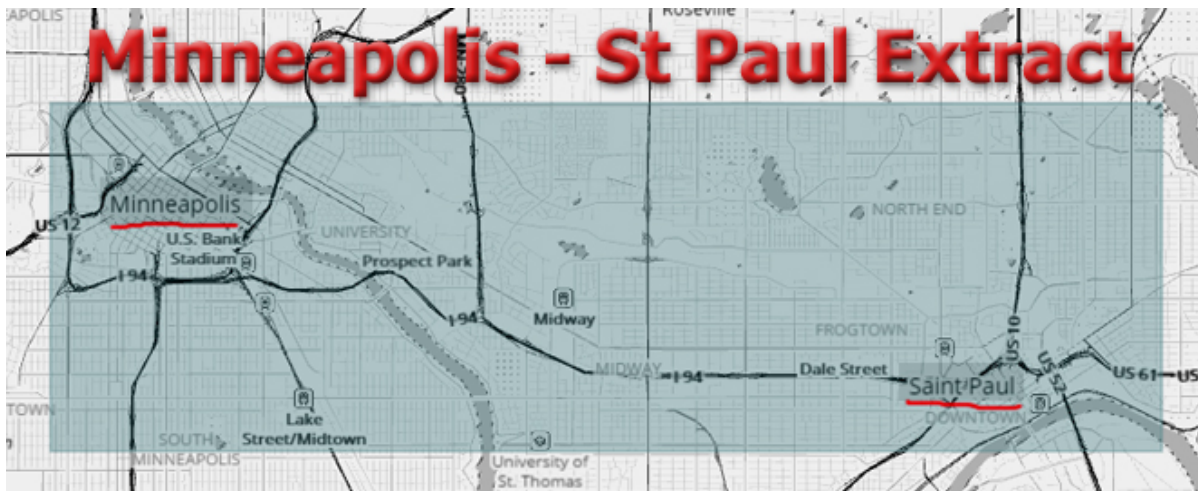


Table of Contents

- Introduction.
- Problems Encountered in the Map.
- DataSet Overview.
- Additional Ideas for Improvements.
- MongoDB Queries and Results.
- Conclusions.

Introduction [TOC] [Top]













OpenStreetMap (<https://www.openstreetmap.org>) is a wonderful opensource street mapping project that is populated by volunteers around the globe. **It does have one problem though**, since it's populated by volunteers, it's data has a quality controll problem. That makes it an **excellent prospect for real world Data Wrangling**.

This project will:

- **Procure data extracts** from the OpenStreetMap project..
- **Assess the quality of the data** for validity, accuracy, completeness, consistency and uniformity.
- **Parse and gather data** from popular file formats such as .json, .xml, .csv, .html.
- **Process data** from many files and very large files that will be cleaned programatically.
- **Store, query, and aggregate data** using MongoDB.

Behind the Scenes Efforts

What is not obvious in this report is that **data procurement** and **coding effort** behind the scenes. The final data extraction ("**Minneapolis_St-Paul.osm**") is the result of multiple attempts with **MapZen** (<https://mapzen.com/data/metro-extracts/>) to procure just the right dataset with a rich enough data representation to be interesting. There are also five **python scripts** which effectively represent the process of auditing, cleaning, transforming and preparing the data to a form in which a local MongoDB instance can readily import it. Finally a text file containing the various **MongoDB Queries** is present that contains the key queries used in the project.

<input type="checkbox"/> Name	Date modified	Type	Size
 Minneapolis_St-Paul.osm.bz2	8/25/2016 1:43 PM	BZ2 File	5,071 KB
 example.osm.json ✓ Examples	8/26/2016 9:55 PM	JSON File	9 KB
 Minneapolis_St-Paul.osm.json	8/28/2016 6:10 PM	JSON File	82,014 KB
 class_example.osm Examples	8/19/2016 9:24 AM	OSM File	7 KB
 example.osm ✓ Examples	8/19/2016 9:24 AM	OSM File	7 KB
 Minneapolis_St-Paul.osm	8/25/2016 1:43 PM	OSM File	72,008 KB
 AUDIT-improving_street_names.py	8/27/2016 11:09 AM	PY File	6 KB
 DATA-prepare_for_mongodb_insertion.py	8/28/2016 6:10 PM	PY File	17 KB
 MAPPARSER-iterative_parsing.py	8/27/2016 11:19 AM	PY File	3 KB
 TAGS-tag_types.py	8/27/2016 10:13 AM	PY File	4 KB
 USERS-exploring_users.py	8/27/2016 10:13 AM	PY File	2 KB
→  MongoDB_Queries.txt MongoDB Queries	8/30/2016 7:36 AM	Text Document	11 KB

Problems Encountered in the Map [Toc] [Top]



Some of the problems I had encountered in my extracted map were the following:

- Incorrect and inconsistent postal codes
- Highly abbreviated street names
- Uneven geographical contributions

Incorrect and inconsistent postal codes

A cursory review uncovered that the postal codes varied between a five number code and a ten number code, which was later trimmed down to a standard 5 digit code for consistency sake.

```

> db.minneapolis.aggregate([
...   {
...     $match: {
...       'address.postcode' : { $ne : null }
...     },
...   },
...   { $unwind: "$address.postcode" },
...   {
...     $group: {
...       _id: { $toLowerCase: '$address.postcode' },
...       count: { $sum: 1 }
...     },
...   },
...   { $sort: { count: -1 } },
...   { $limit: 100 }
... ])
< "_id" : "55414", "count" : 427 >
< "_id" : "55108", "count" : 283 >
< "_id" : "55455", "count" : 162 >
< "_id" : "55104", "count" : 74 >
< "_id" : "55404", "count" : 62 >
< "_id" : "55406", "count" : 58 >
< "_id" : "55101", "count" : 45 >
< "_id" : "55403", "count" : 44 >
< "_id" : "55401", "count" : 43 >
< "_id" : "55454", "count" : 41 >
< "_id" : "55415", "count" : 32 >
< "_id" : "55114", "count" : 31 >
< "_id" : "55408", "count" : 26 >
< "_id" : "55102", "count" : 22 >
< "_id" : "55407", "count" : 21 >
< "_id" : "55402", "count" : 21 >
< "_id" : "55130", "count" : 18 >
< "_id" : "55405", "count" : 10 >
< "_id" : "55106", "count" : 9 >
< "_id" : "55413", "count" : 6 >
Type "it" for more
> it
< "_id" : "55103", "count" : 6 >
< "_id" : "55455-0153", "count" : 5 >
< "_id" : "55108-1013", "count" : 3 >
< "_id" : "55414-2820", "count" : 3 >
< "_id" : "55455-0367", "count" : 3 >
< "_id" : "55108-1036", "count" : 3 >
< "_id" : "55114-1251", "count" : 2 >
< "_id" : "55108-1003", "count" : 2 >
< "_id" : "55108-1046", "count" : 2 >
< "_id" : "55108-6100", "count" : 2 >
< "_id" : "55113", "count" : 2 >
< "_id" : "55414-3063", "count" : 2 >
< "_id" : "55414-3026", "count" : 2 >
< "_id" : "55455-3002", "count" : 2 >
< "_id" : "55108-1015", "count" : 1 >
< "_id" : "55488", "count" : 1 >
< "_id" : "55454-1313", "count" : 1 >
< "_id" : "55414-3528", "count" : 1 >
< "_id" : "55108-1014", "count" : 1 >
< "_id" : "55414-2921", "count" : 1 >
Type "it" for more

```

5 Digit Codes

Ten Digit Codes

Highly abbreviated street names

A quick perusal of the street names revealed that a problem of an overly abbreviated naming convention existed and needed to be rectified. Street names were expanded prior to exporting to the final JSON format (for MongoDB Input).

```

'Clarence Avenue Southeast',
'Franklin Avenue Southeast',
'Pillsbury Drive Southeast',
'Pleasant Street Southeast',
'University Avenue Southeast'},
'St': {'Arlington St',
'Curfew St',
'E 10th St',
'Eustis St',
'Fulham St',
'N 4th St',
'Prince St',
'S 4th St',
'SE Ontario St'},
'Terrace': {'Franklin Terrace', 'Groveland Terrace'},
'W': {'7th Place W',
'Larpenteur Ave W',
'Larpenteur Avenue W',
'University Ave W'},
'West': {'5th St West',
'7th Street West',
'Bandana Boulevard West',
'Central Avenue West',
'Hoyt Avenue West',

```

Abbreviated Street Names

Uneven geographical contributions

Once the Data was in JSON format and ready for insertion into MongoDB, an audit of the data made it clear that there was not sufficient or consistent coverage of the POS (positional data) for many of the objects in the dataset. This was not a fixible problem at this point.

DataSet Overview [\[TOC\]](#) [\[Top\]](#)



A cursory exploration of the dataset is performed highlighting a few key metrics and the MongoDB Queries used to achieve them:

Number of Documents

QUERY:

```
// Count the documents in a collection
db.minneapolis.aggregate(
  [
    { $group: { _id: '_', count: { $sum: 1 } } } ]
)
```

RESULT:

```
{ "_id" : "_", "count" : 378146 }
```

QUERY:

```
// Cross check - Count the documents in a collection
db.minneapolis.find().count()
```

RESULT:

```
378146
```

Size of Data Extraction File

Actual: (86MB)

```
db.minneapolis.stats({ scale : 1048576 })
```

Note: 1048576 is 1MB in Bytes (Base unit of calculation)

```
{
  "ns" : "examples.minneapolis",
  "count" : 378146,
  "size" : 86,
  "avgObjSize" : 238,
  "storageSize" : 26,
  "capped" : false,
  "wiredTiger" : {
    ....
    ....
  }
}
```

Number of Nodes

QUERY:

```
// Count the type = 'nodes' in a collection using aggregation
db.minneapolis.aggregate(
  [
    { $match : { type : 'node' } },
    { $group: { _id: '_', nodesCount: { $sum: 1 } } }
  ]
)
```

RESULT:

```
{ "_id" : "_", "nodesCount" : 324406 }
```

|

QUERY:

```
// Cross check using straight system call
db.minneapolis.count( { type: 'node' } )
```

RESULT:

```
324406
```

Number of Ways

QUERY:

```
// Count the type = 'ways' in a collection using aggregation
db.minneapolis.aggregate(
  [
    { $match : { type : 'way' } },
    { $group: { _id: '_', waysCount: { $sum: 1 } } }
  ]
)
```

RESULT:

```
{ "_id" : "_", "waysCount" : 52024 }
```

QUERY:

```
// Cross check using straight system call
db.minneapolis.count( { type: 'way' } )
```

RESULT:

```
52024
```

Number of Unique Users

QUERY:

```
// Aggregation / pipeline method to calculate the unique users contributing to this extraction:
db.minneapolis.aggregate([
// First group the unique users into a set
{ $group: { _id: { user : '$created.user'}, userSet: { $addToSet: '$created.user' } }},
// Unwind the set holding the unique users
{ $unwind: "$userSet" },
// then count them
{ $group: { _id: "_", uniqueUserCount: { $sum:1 } } }
])
```

RESULT:

```
{ "_id" : "_", "uniqueUserCount" : 466 }
```

QUERY:

```
// Single system call method:
db.minneapolis.distinct('created.user').length
```

RESULT:

```
446
```

Top contributing users**QUERY:**

```
// Top contributing User(s)
db.minneapolis.aggregate([
  { $group:{_id : '$created.user', count :{ $sum : 1 } }},
  { $sort : { count : -1 } },
  { $limit : 20 }
])
```

RESULT:

```
{ "_id" : "Mulad", "count" : 252773 }
{ "_id" : "iandees", "count" : 39129 }
{ "_id" : "neuhausr", "count" : 11924 }
{ "_id" : "stucki1", "count" : 9525 }
{ "_id" : "nickrosencrans", "count" : 8838 }
{ "_id" : "sota767", "count" : 6823 }
{ "_id" : "Mink", "count" : 5753 }
{ "_id" : "mcguirrm", "count" : 5513 }
{ "_id" : "lizzz_msp", "count" : 3834 }
{ "_id" : "Falcorian", "count" : 3144 }
{ "_id" : "woodpeck_fixbot", "count" : 3077 }
{ "_id" : "adamjgardner", "count" : 1655 }
{ "_id" : "DavidF", "count" : 1586 }
{ "_id" : "Mike Dolbow", "count" : 1305 }
{ "_id" : "noliver", "count" : 1181 }
{ "_id" : "PrometheusAvV", "count" : 1169 }
{ "_id" : "Joel_K", "count" : 1137 }
{ "_id" : "WernerP", "count" : 1082 }
{ "_id" : "bot-mode", "count" : 993 }
{ "_id" : "headwatersolver", "count" : 957 }
```

Users with only one post

QUERY:

```
// Number of users having only 1 post
db.minneapolis.aggregate([
  {$group:{_id:'$created.user', count:{$sum:1}}},
  {$group:{_id:'$count', num_users:{$sum:1}}},
  {$sort:{_id:1}}, {$limit:1}
])
```

RESULT:

```
{ "_id" : 1, "num_users" : 121 }
```

Number of chosen types (cafes, restaurants, etc)

• MICROBREWERY'S

QUERY:

```
// Fetch the count of the Pubs which are also $microbrewery's
db.minneapolis.aggregate([
  { $match: {
    amenity : { $exists :1},
    amenity : 'pub' } },
  { $group: { _id : '$microbrewery',
    count: { $sum :1 } }},
  { $sort : { count : -1 } },
  { $limit : 10 }
])
```

RESULT:

```
{ "_id" : null, "count" : 51 }
{ "_id" : "yes", "count" : 9 }
```

QUERY:

```
// List the Pubs which are also $microbrewery's
db.minneapolis.aggregate([
  { $match: {
    amenity : { $exists :1},
    amenity : 'pub',
    microbrewery : 'yes' } },
  { $group: { _id : '$name',
    count: { $sum :1 } }},
  { $sort : { count : -1 } },
  { $limit : 10 }
])
```

RESULT:

```
{ "_id" : "LynLake Brewery", "count" : 1 }
{ "_id" : "The Herkimer Pub & Brewery", "count" : 1 }
{ "_id" : "Surly Brewery", "count" : 1 }
{ "_id" : "Fulton Brewery Tap Room", "count" : 1 }
{ "_id" : "Town Hall Brewery", "count" : 1 }
{ "_id" : "Day Block Brewing Company", "count" : 1 }
{ "_id" : "Tin Whiskers Brewing Company", "count" : 1 }
{ "_id" : "Harriet Brewing", "count" : 1 }
{ "_id" : "Bang Brewing", "count" : 1 }
```

• CUISINE

QUERY:

```
// Fetch the count of the most popular type of food establishment:
db.minneapolis.aggregate([
  { $match: {
    amenity: { $exists :1},
    amenity : 'restaurant' } },
  { $group: { _id : '$cuisine',
    count: { $sum: 1 } }},
  { $sort : { count : -1 } },
  { $limit : 10 }
])
```

RESULT:

```
{ "_id" : null, "count" : 112 }
{ "id" : "pizza", "count" : 16 }
```

```
{ "_id" : "chinese", "count" : 12 }
{ "_id" : "vietnamese", "count" : 12 }
{ "_id" : "italian", "count" : 11 }
{ "_id" : "mexican", "count" : 9 }
{ "_id" : "american", "count" : 8 }
{ "_id" : "thai", "count" : 5 }
{ "_id" : "regional", "count" : 5 }
{ "_id" : "japanese", "count" : 4 }
```

• RELIGION

QUERY:

```
// Fetch the count of each unique religion:
```

```
db.minneapolis.aggregate([
  {
    $match: {
      amenity: {$ne : null},
      amenity: 'place_of_worship'
    }
  },
  { $unwind: "$amenity" },
  {
    $group: {
      _id: {$toLower: '$religion'},
      count: { $sum: 1 }
    }
  },
  { $sort : { count : -1 } },
  { $limit : 10 }
])
```

RESULT:

```
{ "_id" : "christian", "count" : 102 }
{ "_id" : "", "count" : 17 }
{ "_id" : "muslim", "count" : 3 }
{ "_id" : "unitarian_universalist", "count" : 2 }
{ "_id" : "jewish", "count" : 1 }
{ "_id" : "buddhist", "count" : 1 }
{ "_id" : "harddeism", "count" : 1 }
```

Additional Ideas for Improvements [TOC] [Top]

Contributor Skewness

The simple statistics below show a most egregious skewness of the contributions to the extraction dataset. One user contributing **68.8%** of the overall submissions is far too concentrated to tolerate. There must be better way to motivate users to contribute other than monetary rewards. Such ways might come in the form of recognition both within the user community and outside (Stack Overflow, Quora, Media [Online, Radio, etc...],

and the like).

- **Skewness of Contributors**

- Number of total documents: **378,146** ✓
- Number of unique users: **446** ✓

Top Contributors ranked in order of submissions:

1) { "_id" : "Mulad", "count" : 252773 }
TOTAL: 252,773 = .668 % [**1 user**] of Total Users (1/446)

2) { "_id" : "iandees", "count" : 39129 }
3) { "_id" : "neuhausr", "count" : 11924 }
TOTAL: 303,826 = .803 % [**< 1 %**] of Total Users (3/446)

4) { "_id" : "stucki1", "count" : 9525 }
5) { "_id" : "nickrosencrans", "count" : 8838 }
6) { "_id" : "sota767", "count" : 6823 }
7) { "_id" : "Mink", "count" : 5753 }
8) { "_id" : "mcguirrm", "count" : 5513 }
SUB: 36,452
TOTAL 340,278 = .899 % [**< 1.8 %**] of Total Users (8/446)

9) { "_id" : "lizzz_msp", "count" : 3834 }
10) { "_id" : "Falcorian", "count" : 3144 }
11) { "_id" : "woodpeck_fixbot", "count" : 3077 }
SUB: 10,055
TOTAL: 350,333 = .926 % [**< 2.4 %**] of Total Users (11/446)

GeoPositional Data Coverage

The **POS** data does not sufficiently cover all of the map extraction (however it **does** cover the parts it intends to cover. I'd recommend partnering users with geocaching expeditions and groups to foster a secondary goal to establish geoposition data in areas which are not represented just yet.

Benefits and Problems

The benefits of these suggestions are potentially: (1) better distribution of users contributing to the OSM dataset across the board, and (2) a start at getting better coverage of geopositional data for existing coverage areas.

The downside to these suggestions are (like anything in the behavioral sciences): (A) there is no guarantee their implementation will actually translate into the envisioned result, (b) the cost of implementing these schemes might prove too high to warrant their existence, and (3) a diluted implementation might not have the potency of it's full strength counterpart.

Additional Ad Hoc Statistics and Queries [TOC] [Top]



Additional Ad Hoc Statistics and Queries

- **RAILWAY**

QUERY:

```
// Find the number of distinct railways using a system call
db.minneapolis.distinct('railway').length
```

RESULT:

```
17
|
```

- **BUILDING**

QUERY:

```
// Find the number of distinct building using a system call
db.minneapolis.distinct('building').length
```

RESULT:

```
41
```

- **NATURAL**

QUERY:

```
// Find the number of distinct natural elements using a system call
db.minneapolis.distinct('natural').length
```


RESULT:

```
9
```

- **LIST NATURAL ELEMENTS**

QUERY:

```
// List the unique natural elements and their individual counts using aggregation
db.minneapolis.aggregate([
  {
    $match: {
      natural: {$ne : null}
    }
  },
  { $unwind: "$natural" },
  {
    $group: {
      _id: {$toLower: '$natural'},
      count: { $sum: 1 }
    }
  },
  { $sort: { count: -1 } },
  { $limit: 100 }
])
```

RESULT:


```
{ "_id" : "tree", "count" : 2967 }
{ "_id" : "water", "count" : 69 }
{ "_id" : "tree_row", "count" : 35 }
{ "_id" : "sand", "count" : 31 }
{ "_id" : "wood", "count" : 7 }
{ "_id" : "wetland", "count" : 7 }
{ "_id" : "cliff", "count" : 4 }
{ "_id" : "beach", "count" : 2 }
{ "_id" : "bay", "count" : 1 }
```

• LIST BUILDINGS TYPES AND COUNTS

QUERY:

```
// List the unique buildings and their individual counts using aggregation
db.minneapolis.aggregate([
  {
    $match: {
      building: {$ne : null}
    }
  },
  { $unwind: "$building" },
  {
    $group: {
      _id: {$toLower: '$building'},
      count: { $sum: 1 }
    }
  },
  { $sort : { count : -1 } },
  { $limit : 20 }
])
```

RESULT:



```
{ "_id" : "yes", "count" : 12319 }
{ "_id" : "house", "count" : 8221 }
{ "_id" : "garage", "count" : 5699 }
{ "_id" : "apartments", "count" : 603 }
{ "_id" : "residential", "count" : 253 }
{ "_id" : "shed", "count" : 226 }
{ "_id" : "garages", "count" : 112 }
{ "_id" : "roof", "count" : 98 }
{ "_id" : "commercial", "count" : 45 }
{ "_id" : "office", "count" : 28 }
{ "_id" : "school", "count" : 18 }
{ "_id" : "church", "count" : 16 }
{ "_id" : "retail", "count" : 16 }
{ "_id" : "industrial", "count" : 14 }
{ "_id" : "public", "count" : 14 }
{ "_id" : "university", "count" : 10 }
{ "_id" : "civic", "count" : 9 }
{ "_id" : "dormitory", "count" : 7 }
{ "_id" : "hospital", "count" : 5 }
{ "_id" : "duplex", "count" : 4 }
```


• LIST RAILWAY TYPES AND COUNTS

QUERY:

// List the unique railways and their individual counts using aggregation

```
db.minneapolis.aggregate([
  {
    $match: {
      railway: {$ne : null}
    }
  },
  { $unwind: "$railway" },
  {
    $group: {
      _id: {$toLower: '$railway'},
      count: { $sum: 1 }
    }
  },
  { $sort : { count : -1 } },
  { $limit : 20 }
])
```

RESULT:



```
{ "_id" : "rail", "count" : 512 }
{ "_id" : "level_crossing", "count" : 320 }
{ "_id" : "crossing", "count" : 266 }
{ "_id" : "switch", "count" : 216 }
{ "_id" : "light_rail", "count" : 179 }
{ "_id" : "abandoned", "count" : 123 }
{ "_id" : "platform", "count" : 51 }
{ "_id" : "station", "count" : 31 }
{ "_id" : "disused", "count" : 25 }
{ "_id" : "preserved", "count" : 18 }
{ "_id" : "railway_crossing", "count" : 11 }
{ "_id" : "buffer_stop", "count" : 10 }
{ "_id" : "signal", "count" : 10 }
{ "_id" : "dismantled", "count" : 5 }
{ "_id" : "proposed", "count" : 5 }
{ "_id" : "turntable", "count" : 2 }
{ "_id" : "construction", "count" : 1 }
```

Conclusion [TOC] [Top]



Thoughts and Perspectives

Aside from commenting on the quality of the extraction used in this project (which was **quite good actually**, all things considered), the key question is how good is open source OSM type map data compared to its commercial counterparts? This is a topic of significant interest (see research papers listed below) (GIS Stack Exchange) from the **academic perspective** (How good is Volunteered Geographical Information?) and **commercial** (OSM Founder: OpenStreetMap already as good or better than google maps.). The general answer (and one which this project supports), is that OSM data (despite it's flaws) is pretty good, is closing the gap with it's commercial counterparts, and in some specific cases, actually better.



See this [paper](#) (draft version):

18



Haklay, M. (2010), How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets. *Environment and Planning B: Planning and Design* 37(4) 682 – 703.



for more rigorous assessment (in UK context).

[This](#) one for assesment (in comparison with Google Maps & Bing Maps) in Ireland:

Ciepluch, B., Jacob, R., Mooney, P., Winstanley, A. (2010) Comparison of the accuracy of OpenStreetMap for Ireland with Google Maps and Bing Maps. *Proceedings of the Ninth International Symposium on Spatial Accuracy Assessment in Natural Resources and Environmental Sciences* 20-23rd July 2010, p. 337.

And [this](#) for assessment in France (paywall):

Girres, J.-F. & Touya, G. (2010) Quality Assessment of the French OpenStreetMap Dataset. *Transactions in GIS*, 14, 435-459.

Germany is covered in this [paper](#):

Zielstra, D., Zipf, A. (2010) Quantitative Studies on the Data Quality of OpenStreetMap in Germany. Paper presented at GIScience 2010.

Very thorough comparison with TomTom data (in Germany as well) is covered [here](#):

Helbich, M., Amelunxen, C., Neis, P. (2012): *Comparative Spatial Analysis of Positional Accuracy of OpenStreetMap and Proprietary Geodata*. Int. GI_Forum 2012. Salzburg, Austria.

German street network is also discussed in detail [here](#):

Neis, P.; Zielstra, D.; Zipf, A. (2012) The Street Network Evolution of Crowdsourced Maps: OpenStreetMap in Germany 2007–2011. *Future Internet* 4, 1-21.

There also two Master's dissertations dealing with this topic: [one](#) from UCL:

Kounadi, O. (2009) Assessing the quality of OpenStreetMap data.

and [one](#) from UNIGIS:

Stark, H.-J. (2010) Quality assurance of crowdsourced geocoded address-data within OpenAddresses Concepts and Implementation

Also [article](#) from Cartographica (paywall) might be of interest:

Mondzech, J. & Sester, M., (2011) Quality Analysis of OpenStreetMap Data Based on Application Needs. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 46, 115-125.