Assessing the Completeness of Geographical Data

Simon Razniewski and Werner Nutt

Free University of Bozen-Bolzano
{razniewski,nutt}@inf.unibz.it

Abstract. Geographical databases are often incomplete, especially when built up incrementally and by volunteers. A prominent example is OpenStreetMap. Often such databases contain also metadata saying that certain features are completely captured for certain areas. We show how to use such metadata to analyse in which areas queries return a complete answer. Such "completeness areas" can be computed via standard spatial operations. Still larger completeness areas can be derived if not only metadata but also the actual content of the database is taken into account. Finally, we discuss which challenges arise if one wants to practically utilize the completeness metadata in OpenStreetMap.

1 Introduction

Storage and querying of geographic information has always been important. Recently, due to the increased availability of GPS devices, volunteered geographical information systems, in particular OpenStreetMap, have quickly evolved. Ongoing open public data initiatives that allow to integrate government data also contribute. The level of detail of OpenStreetMap is generally significantly higher than that of commercial solutions such as Google Maps or Bing Maps, while its accuracy and completeness are comparable.

OpenStreetMap (OSM) allows to collect information about the world in remarkable detail. This, together with the fact that the data is collected in a voluntary, possibly not systematic manner, brings up the question of the completeness of the OSM data. When using OSM, it is desirable also to get metadata about the completeness of the presented data, in order to properly understand its usefulness.

Judging completeness by comparing with other data is only possible, if more reliable data exists, which is generally not the case. Therefore, completeness can best be assessed by metadata about the completeness of the data, that is produced in parallel to the base data, and that can be compiled and shown to users. In geographical database systems it is very common to collect metadata, as widespread standards such as the FGDC metadata standard show. However, little is known about how query answers can be annotated with completeness information.

As an example, consider that a tourist wants to find hotels in some town that are no further than 500 meters away from a spa. Assume, that, as shown in Figure 1, the data about hotels and spas is only complete in parts of the map. Then, the query answer is only complete in the intersection of the areas where hotels are complete and a zone 500 meters inside the area where spas are complete (green in the figure), because outside, either hotels or spas within 500 meters from a hotel could be missing from the database, thus leading to missing query results.

D. Olteanu, G. Gottlob, and C. Schallhart (Eds.): BNCOD 2013, LNCS 7968, pp. 228–237, 2013.
© Springer-Verlag Berlin Heidelberg 2013

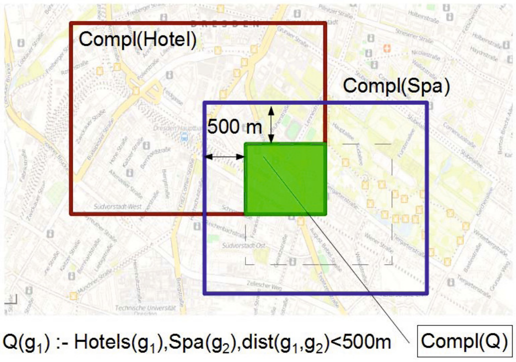


Fig. 1. Spatial query completeness analysis example. For a more complex one, see Figure 3.

Our contribution in this paper is a general solution for reasoning about the completeness of spatial data. In particular, we show that metadata can allow elaborate conclusions about query completeness, when one takes into account the data actually stored in the database. We also show that metadata about completeness is already present to a limited extent for OSM, and discuss practical challenges regarding acquisition and usage of completeness metadata in the OSM project.

The structure of this paper is as follows: In Section 2, we discuss spatial database systems, online map services, geographical data completeness and OpenStreetMap. In Section 3, we give necessary formalizations to discuss the problem of reasoning about geographical data completeness in Section 4. In Section 5, we discuss practical issues regarding the applicability of our solution to OpenStreetMap.

2 Background

2.1 Spatial Databases Systems and Online Map Services

To facilitate storage and retrieval, geographic data is usually stored in spatial databases. According to [1], spatial databases have three distinctive features. First, they are database systems, thus classical relational/tree-shaped data can be stored in them and retrieved via standard database query languages. Second, they offer spatial data types, which are essential to describe spatial objects. Third, they efficiently support spatial data types via spatial indexes and spatial joins.

Online map services usually provide graphical access to spatial databases and provide services for routing and address finding. There are several online map services available, some of the most popular ones being Google Maps, Bing Maps, MapQuest and OpenStreetMap. With the exception of OSM, the data underlying those services is not freely accessible. The most common uses of those services are routing ("Best path from A to B?"), address retrieval ("Where is 2nd street?") and business retrieval ("Hotels in Miami"). While the query capabilities of most online map services are currently still limited (one can usually only search for strings and select categories), spatial databases generally allow much more complex queries.

Example 1. Tourists could be interested in finding those hotels that are less than 500 meters from a spa and 1 kilometer from the city center. Real estate agents could be interested in properties that are larger than 1000 square meters and not more than 5 kilometers from the next town with a school and a supermarket. Evacuation planners might want to know which public facilities (schools, retirement homes, kindergartens, etc.) are within a certain range around a chemical industry complex.

2.2 Geographical Data Completeness

In addition to precision and accuracy, knowledge about completeness is essential when using spatial data [2]. Completeness describes the extent to which features in the real world that are of interest are also present in the database. Completeness can highly vary for different features. If metadata about completeness is present, it is attractive to visualize it on maps [3]. Completeness is especially a challenge when (1) databases are to capture continuously the current state (as opposed to a database that stores a map for a fixed date) because new features can appear, (2) databases are built up incrementally and are accessible during build-up (as it is the case for OSM) and (3) the level of detail that can be stored in the database is high (as it is easier to be complete for all highways in a state than for all post boxes).

2.3 OpenStreetMap

OpenStreetMap is a wiki-style project for building a map of the world. Contributors to the project are volunteers that commonly use GPS devices to track paths and features or use aerial pictures to identify them. In contrast to most commercial products, the project's data is freely available and map data can be reused in other applications and visualized for different specific needs. In most areas of the world, OSM provides significantly more information than its commercial competitors.[1] The level of detail of features that can be stored is remarkable, as for example for buildings, entrances and building heights can be stored, for vending machines the kind of good they sell or for waste baskets the kind of waste they accept.

There have been some attempts at formalizing [4] and assessing the quality of OSM [5,6]. The latter showed that the road map coverage of England is quite good, but it is clear that, because of the level of detail that can be stored, many non-core features are currently far from being complete.

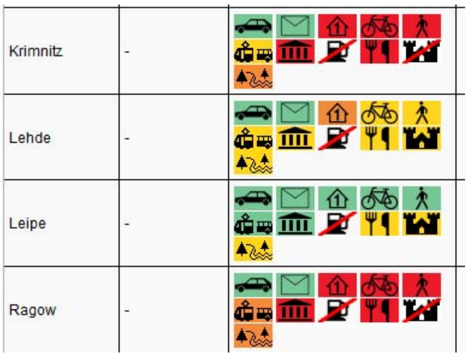
Assessment of completeness is not only important for consumers but also for the producers of the data (mappers), because they are interested to know where to direct their work. Some OSM mappers therefore introduced the idea of storing information about completeness of different features in tables on the OSM wiki [7]. An example of such a table is shown in Figure 2.

3 Formalization

3.1 Spatial Databases

A *geometry* is a datatype used to represent the location and extent of a spatial object, for example as a point or a polygon.

[1] For a graphical comparison tool, see for example `http://tools.geofabrik.de/mc/`.



Krimnitz	-	🟩🟩🟥🟥🟥 🟨🟥⊘🟥⊘ 🟧
Lehde	-	🟩🟩🟧🟨🟨 🟨🟨⊘🟨🟨 🟨
Leipe	-	🟩🟩🟩🟩🟩 🟩🟩⊘🟨🟨 🟨
Ragow	-	🟩🟩🟥🟥🟥 🟧🟥⊘🟥⊘ 🟧

Fig. 2. Completeness information for different districts of Lübbenau, Germany. Each symbol stands for a set of features, the colors for its completeness level. Taken from [8].

We assume that a *spatial database* consists of a set of relations, where each relation has besides other attributes exactly one attribute g of type geometry.

We consider a special class of queries, which we call star-shaped queries, that allow only joins over the distance-less-than relation with the output relation. Formally, a *star-shaped query* is written as

$$\begin{aligned} Q(g_0) :- R_0(\bar{x}_0, g_0), M_0, R_1(\bar{x}_1, g_1), M_1, \mathrm{dist}(g_0, g_1) < c_1, \ldots, \\ R_n(\bar{x}_n, g_n), M_n, \mathrm{dist}(g_0, g_n) < c_n, \end{aligned} \tag{1}$$

where each M_i is a set of comparisons of the arguments of R_i with constants.

Example 2. Consider again the query asking for hotels that are closer than 500 meters to a spa and closer than 1 km to the city center. This can be written as

$$\begin{aligned} Q_{\text{hotels}}(g_0) :- \ \mathrm{Hotel}(n, r, g_0), \mathrm{Spa}(g_1), \mathrm{dist}(g_0, g_1) < 500, \\ \mathrm{Center}(g_2), \mathrm{dist}(g_0, g_2) < 1000, \end{aligned}$$

if we assume a ternary relation Hotel with attributes name, rating, and geometry and unary relations for spas and centers. The query is star-shaped, because both spa and center are joined with hotel, which is the relation providing the output. The other queries in Example 1 are star-shaped as well.

3.2 Completeness

Real-world databases are often incomplete. This was formalized by Motro [9] such that incomplete databases are actually pairs (D^i, D^a), where the *available* (real) database D^a contains only a subset of the facts that hold in the *ideal* database D^i, which represents the complete information about the world.

Example 3. Consider that in the real world, there are a Hilton (4 stars), a Sheraton (5 stars) and a Best Western (3 stars) hotel in a town, but in the database, the information about the Best Western is missing. Ideal and available database for that scenario are shown in Table 1.

Table 1. Example of an ideal and available database

Ideal database — Relation Hotel		
name	rating (stars)	geometry
Hilton	4	P(46.61, 12.30)
Sheraton	5	P(46.62, 12.30)
Best Western	3	P(46.64, 12.26)

Available database — Relation Hotel		
name	rating (stars)	geometry
Hilton	4	P(46.61, 12.30)
Sheration	5	P(46.62, 12.30)

To express partial completeness, Levy introduced a format for metadata about database completeness, which he called local completeness statements [10]. We extend those to feature completeness statements. Formally, a *feature completeness statement* F is written as $\text{Compl}(R(\bar{x}, g); M; A)$ and consists of a relation name R, a set M of comparisons of the attributes of R with constants, and an area A. The feature completeness statement holds over a pair of an ideal and an available database (D^i, D^a), if its associated query $Q_F(g) :\!- R(\bar{x}, g), M$ returns the same answers over both databases in the area A, that is, if $Q_F(D^i) \cap A = Q_F(D^a) \cap A$.

Example 4. The database in Table 1 is complete for all hotels with more than 3 stars on the full map, that is, it satisfies the statement $F_1 = \text{Compl}(\text{Hotel}(n, r, g); \{r > 3\}; \mathit{FULL\_MAP})$. Furthermore, it is complete for all hotels in areas south of 46.63 latitude, that is, it satisfies $F_2 = \text{Compl}(\text{Hotel}(n, r, g); \emptyset; \mathit{RECTANGLE}(0,\ 0\ \ 46.63,\ 20))$.

Let $\mathcal{F}$ be a set of feature completeness statements $\mathcal{F}$ and Q be a star-shaped query. We define the completeness area CA of Q wrt $\mathcal{F}$ as the set of points p such that for all pairs of an ideal and an available database (D^i, D^a) that satisfy $\mathcal{F}$, it holds that $p \in Q(D^i)$ implies $p \in Q(D^a)$. In other words, Q will not miss p as an answer over the available database if it would return p over the ideal database. Thus, CA is the largest area such that $Q(D^i) \cap CA = Q(D^a) \cap CA$ for all (D^i, D^a) satisfying $\mathcal{F}$.

Example 5. Consider the query $Q_{\text{FiveStars}}(g) :\!- \text{Hotel}(n, r, g), r = 5$ that asks for all hotels with five stars. Assuming only the feature completeness statement F_1 from above holds, CA is FULL_MAP. Assuming, however, only the statement F_2 holds, CA is RECTANGLE(0, 0 46.63, 20).

4 Query Completeness Analysis

Given a query Q and a set of FC statements $\mathcal{F}$, the completeness analysis problem is to determine CA. We first analyse the general problem, then show that larger completeness areas can be derived, if the available database is taken into account. For simplicity, we assume in the following that the geometries of all features are points, and consider only star-shaped queries.

4.1 Analysis without the Database Instance

We first consider completeness analysis problem for *simple* queries, which are of the form $Q(g) :\!- R(\bar{x}, g), M$.

Let $\mathcal{F}$ be a set of FC statements and R be a relation. Then we denote by $\mathcal{F}^R$ the subset of $\mathcal{F}$ that consists of those statements that talk about R. Suppose $F_1, \ldots, F_m$ is an enumeration of the statements in $\mathcal{F}^R$. Then each F_j in $\mathcal{F}^R$ is of the form Compl($R(\bar{x}, g)$; $M_j; A_j$), where M_j is a set of comparisons over the variables in $\bar{x}$ and A_j is an area.

Proposition 1. *Let $\mathcal{F}$ be a set of FC statements and $Q(g) :\!- R(\bar{x}, g), M$ be a simple query. Then CA, the completeness area of Q wrt $\mathcal{F}$, satisfies*

$$CA = \bigcup \{ \textstyle\bigcap_{F_j \in \mathcal{F}_0} A_j \mid \mathcal{F}_0 \subseteq \mathcal{F}^R \text{ and } M \models \bigvee_{F_j \in \mathcal{F}_0} M_j \}.$$

The preceding proposition says that CA is a union of areas $\bigcap_{F_j \in \mathcal{F}_0} A_j$, which are obtained as follows. One chooses a subset $\mathcal{F}_0$ of statements F_j, such that the query condition M entails the disjunction of the conditions M_j of the F_j. Intuitively, this means that the M_j cover all possible ways in which M can be satisfied. Then one intersects the areas A_j of the statements in $\mathcal{F}_0$. Of course, it suffices to take only minimal subsets $\mathcal{F}_0$ of $\mathcal{F}^R$ (wrt set inclusion) whose comparisons are entailed by M.

We remark that the decision version of the problem to determine the CA for a simple query ("Given $\mathcal{F}$, Q and a point p, is p in CA?") is coNP-hard. This can be seen by an encoding the tautology problem for propositional logic. The problem is also in coNP because to show that a point is not in CA, it suffices to guess values for the attributes of R such that none of the conditions of the FC statements that hold in p are satisfied. The hardness however may be practically not very problematic, because (i) it is maybe rather uncommon that FC statements only together imply completeness, because for that, they must complement each other, while it is more realistic, that they only linearily extend each other, and (ii) because geometries of FC statements may seldomly be overlapping, since they are expected to be given for administrative units.

To formulate a characterization of the completeness sets of arbitrary star-shaped queries, we need some additional notation. For an area A and a number $c > 0$, we denote with $\mathit{shrink}(A, c)$ the set of points p in A such that the distance between p and the complement of A is at least c. Intuitively, these are points that are lying deep in A.

For an arbitrary star-shaped query Q as in Equation (1) we introduce $n + 1$ simple queries $Q_0, \ldots, Q_n$, defined as $Q_i(g_i) :\!- R_i(\bar{x}_i), M_i$ for $i = 1, \ldots, n$, which we call the component queries of Q.

Theorem 1. *Let $\mathcal{F}$ be a set of FC statements, $Q(g_0)$ a query as in Equation (1), and $Q_0, \ldots, Q_n$ the component queries of Q. Then CA, the completeness area of Q wrt $\mathcal{F}$, satisfies*

$$CA = CA_0 \cap \mathit{shrink}(CA_1, c_1) \cap \ldots \cap \mathit{shrink}(CA_n, c_n),$$

where CA_i is the completeness area of Q_i.

Example 6. See again Figure 1. There, hotels are complete within the brown rectangle and spas within the blue. The area shrink(Spa, 500) is indicated by the dashed line. Completeness of the query only holds inside the green area, because for any point outside the brown rectangle, there could be a hotel missing in the database that has a spa nearby, and for any point outside shrink(Spa, 500) there could be a spa missing outside the blue rectangle, that indicates that a hotel has the property of having a spa within 500 meters.

Algorithms to compute CA_i and $CA(Q, \mathcal{F})$ are given below:

```
CompleteArea(Feature R, Conditions M, Set of FC statements F)
  Area = empty
  for each subset S of F:
    if statements in S imply completeness of R,M
    then Area = Area.union(intersection of geometries in S)
  return Area
```

```
CalcQueryComplW/oInstance(Query Q, Set of FC statements F)
  Area = CompleteArea($R_0$, $M_0$, F)
  for each pair $R_i$, $M_i$ i>0 in Q:
    AtomArea = ShrinkArea(CompleteArea($R_i$,$M_i$,F),$c_i$)
    Area = Area.intersect(AtomArea)
  return Area
```

Listing 1.1. Algorithms to compute the complete area for an atom and for a query

4.2 Analysis with the Database Instance

When analysing query completeness, one can also take into account the actual state of the database. The problem is then, given a query Q, a set of FC statements $\mathcal{F}$ and an available database instance D^a, to find the largest area CA where it holds for all ideal databases D^i, that, whenever (D^i, D^a) satisfies $\mathcal{F}$, then $Q(D^i) \cap CA = Q(D^a) \cap CA$. Taking into account the available database, more completeness can be derived.

Example 7. Consider Figure 3. The completeness statements are the same as in Figure 1, however now, there are also the positions of the hotels h_1, h_2, h_3 and spas s_1 and s_2 shown, as they are stored in D^a. The area CA is now significantly larger for two reasons:

1. All points within Compl(Hotel), where no hotel is located that could possibly be an answer (i.e., all points except h_2), are now added to CA. The reason is that for all those points it holds that there cannot be any hotel missing that satisfies the query condition, because hotels are complete in that point.
2. All points within *shrink*(Compl(Spa), 500m) where no spa is within a distance of 500m are added, because for those points, although hotels could possibly be missing from the database, none can be missing that satisfies the query condition, because there are no spas around within 500 meters and spas are complete within 500 meters.

To formalize our observations, we first introduce two definitions. Consider a query Q as in Equation (1), a set of FC statements $\mathcal{F}$ and an available database D^a.

We say that a point p is a *potential answer* to Q, if (i) there is an atom $R(\bar{t}, p)$ in $R(D^a)$, (ii) p is not in $Q(D^a)$ and (iii) there exists an ideal database D^i with $(D^i, D^a) \models \mathcal{F}$ such that $p \in Q(D^i)$. We denote the set of all potential answers of Q by $Pot(Q)$. The potential answers can be calculated by removing from $\pi_g(R)$ all those points that are a certain answer (i.e., are in $Q(D^a)$ and that are an impossible answer (i.e., cannot be in $Q(D^i)$

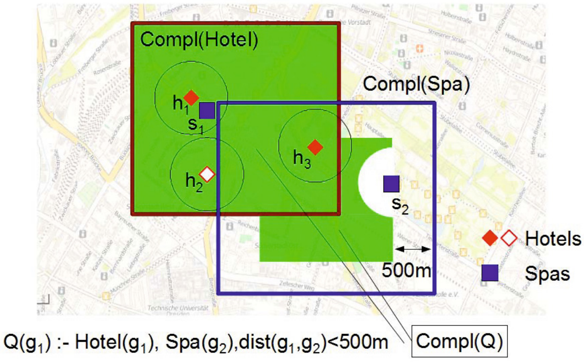


Fig. 3. Completeness analysis when taking into account the database instance. Observe that the point h_2 does not belong to the completeness area.

because some join partner A_i is not in the range c_i in D^a and the atom is also complete in the area buffer(p, c_i), which is the set of points with a distance of at most c_i from p.

For a pair $G_i = R_i(\bar{x}), M_i$ in Q, we define the area $\mathit{ComplOutOfRange}_{\mathcal{F},D^a}(G_i)$ as $\mathit{shrink}(CA_{A_i}, c_i) \setminus \bigcup\{\mathit{buffer}(p, c_i) \mid p \in Q(g) :\!- A_i(\bar{x}, g), M_i\}$.

Now, we can compute the largest area in which a query Q is complete wrt. F and D^a by taking the union of the area where G_0 is complete with the areas where each G_i is complete and out of reach in the distance c_i, and finally removing the points where potential answers are located:

Theorem 2. *Let Q be a star-shaped query with n atoms, F be a set of FC statements and D^a be an available database. Then*

$$CA(Q, F, D^a) \ = \ CA_{A_0} \ \cup \ (\bigcup_{i=1...n} \mathit{ComplOutOfRange}_{\mathcal{F},D^a}(G_i)) \ \setminus \ \pi_g(\mathit{Pot}(Q)).$$

5 Practical Issues in OpenStreetMap

In the OpenStreetMap-wiki, a template for tables that store completeness information exists (see Figure 2). The template introduces 6 levels of completeness (“Largely incomplete” to “Completeness verified by two mappers”) for 11 different features (such as roads or sights), and is used on approximately 1,100 Wiki pages (estimate based on number of pages that contain an image used in the table), which corresponds to 5% of all pages on the Wiki (21,989 content pages on 22.01.2013).

The table in Figure 2 expresses for example that the roads in Lehde are generally complete, the cycling paths largely and the house numbers partially.

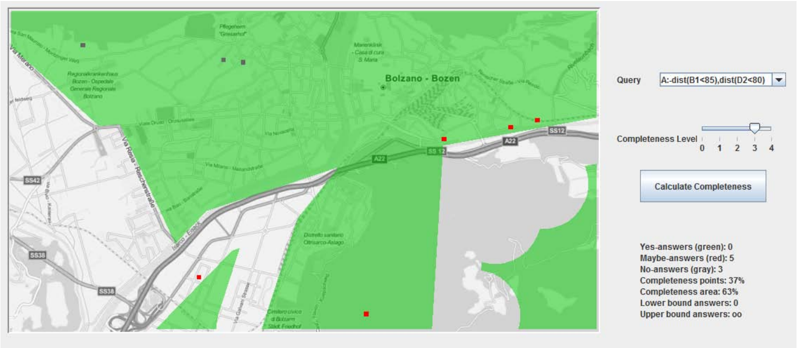
The completeness statements that can be expressed via those tables are of a simpler form than the ones discussed before, because there cannot be any comparisons. Also, there can be at most one completeness statement per relation per area, and the areas are disjoint, since they talk about different administrative units. Altogether, this makes the query completeness analysis computationally easy.

Practically challenging is especially the proper interpretation of those completeness levels: It is hard to say what “Largely complete” can actually mean, and whether that level of completeness is satisfactory for a user’s request. A possible solution would be to use percentages instead of informal descriptions for completeness statements, however the problem then is that giving correct numbers (“60% complete”) is only possible, if one has a good estimate of what 100% would be.

Another serious challenge is to get mappers to widely give those completeness statements. The current usage (5%) is clearly insufficient. A possible reason is that completeness statements introduce a kind of negative responsibility: One states, that there are no unmapped features in a certain kind of area. This is a much stronger statement than saying that a feature is present at some place, which is the usual implicit statement that mappers give when adding a feature to the map.

A third challenge is provided by changes in the real world: While during the build-up of a geographical database changes in the real world are less important, during longer runs, also changes becomes a problem, because features that can have been correct in the past can disappear (e.g. hotels can close), or new features can appear (e.g. new hotels can open). Thus, completeness statements would need periodic review, a possibly not very attractive activity (Wikipedia has a similar problem with its review system).

Minor technical challenges are that the completeness statements would need to be crawled from the OSM-wiki pages, that the district borders used in the Wiki do not always exist in the database and that the OSM data is not stored in relational format but in XML format.



Query A:-dist(B1<85),dist(D2<80)

Completeness Level 0 1 2 3 4

Yes-answers (green): 0
Maybe-answers (red): 5
No-answers (gray): 3
Completeness points: 37%
Completeness area: 63%
Lower bound answers: 0
Upper bound answers: oo

Fig. 4. Screenshot of our demo program for reasoning about the completeness of OSM

6 Conclusion

We have shown that completeness is a major concern in geographical databases that are built up incrementally and by volunteers, and that that holds particularly for OpenStreetMap. We also showed that for analyzing database completeness, metadata is required, because completeness can normally not be analyzed by looking at the data itself.

We showed how metadata about feature completeness can be used to analyze query completeness, and that more completeness can be concluded when not only the metadata but also the database content is taken into account.

We discussed that major challenges for practical applicability are the willingness of mappers to give completeness statements and to review them regularly.

We also implemented a small test program that contains a fictive scenario for the town of Bolzano, which is available under [13]. A screenshot is also shown in Figure 4.

Future work will focus on implementation techniques for a tool that is built upon real OSM data and on discussing the applicability and usefulness of our proposal with the OSM community.

Acknowledgement. We are thankful to the user Bigbug21 for information about the OSM community.

References

1. Güting, R.H.: An introduction to spatial database systems. VLDB J 3(4), 357–399 (1994)
2. Shi, W., Fisher, P., Goodchild, M.: Spatial Data Quality. CRC (2002)
3. Wang, T., Wang, J.: Visualisation of spatial data quality for internet and mobile GIS applications. Journal of Spatial Science 49(1), 97–107 (2004)
4. Mooney, P., Corcoran, P., Winstanley, A.: Towards quality metrics for OpenStreetMap. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 514–517. ACM (2010)
5. Haklay, M., Ellul, C.: Completeness in volunteered geographical information—the evolution of OpenStreetMap coverage in England (2008-2009). Journal of Spatial Information Science (2010)
6. Haklay, M.: How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets. Environment and Planning. B, Planning & Design 37(4), 682 (2010)
7. OSM Wiki, First appearance of completeness statement in OSM (December 2007), http://wiki.openstreetmap.org/wiki/Talk:Landkreis_M%C3%BCnchen
8. OSM Wiki, Completeness statements in the OSM wiki for Luebbenau, http://wiki.openstreetmap.org/wiki/L%C3%BCbbenau
9. Motro, A.: Integrity = Validity + Completeness. ACM Transactions on Database Systems (TODS) 14(4), 480–502 (1989)
10. Levy, A.: Obtaining complete answers from incomplete databases. In: Proc. VLDB, pp. 402–412 (1996), http://portal.acm.org/citation.cfm?id=645922.673332
11. Razniewski, S., Nutt, W.: Completeness of queries over incomplete databases. Proc. VLDB Endow 4(11), 749–760 (2011)
12. Nutt, W., Razniewski, S.: Completeness of queries over sql databases. In: CIKM (2012)
13. Razniewski, S.: Completeness reasoning test program, http://www.inf.unibz.it/~srazniewski/geoCompl/