staty: Quality Assurance for Public Transit Stations in OpenStreetMap (Demo Paper)

Hannah Bast
University of Freiburg
Freiburg, Germany
bast@cs.uni-freiburg.de

Patrick Brosi
University of Freiburg
Freiburg, Germany
brosi@cs.uni-freiburg.de

Markus Näther
University of Freiburg
Freiburg, Germany
naetherm@cs.uni-freiburg.de

ABSTRACT

We present *staty*, a browser-based tool for quality assurance of public transit station tagging in OpenStreetMap (OSM). Building on the results of a similarity classifier for these stations, our tool visualizes name tag errors as well as incorrect and/or missing station group relations. Detailed edit suggestions are provided for individual objects. This is done intrinsically without an external ground truth. Instead, the underlying classifier is trained on the OSM data itself. We describe how our tool derives errors and suggestions from station tag similarities and provide experimental results on the OSM data of the United Kingdom, the United States, and a dataset consisting of Germany, Switzerland, and Austria. Our tool can be accessed under https://staty.cs.uni-freiburg.de.

CCS CONCEPTS

• **Information systems** → **Geographic information systems**; *Web applications*; • **Social and professional topics** → *Quality assurance*;

KEYWORDS

OpenStreetMap Data, Public Transit Data, Quality Assurance

ACM Reference Format:
Hannah Bast, Patrick Brosi, and Markus Näther. 2020. staty: Quality Assurance for Public Transit Stations in OpenStreetMap (Demo Paper). In *28th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '20), November 3–6, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3397536.3422342

1 INTRODUCTION

In OpenStreetMap (OSM), the world is described as a collection of *nodes* (single points on earth), *ways* (lists of nodes, possibly polygonal), and groups thereof, called *relations* (which may contain other relations). All objects can be outfitted with key/value pairs (tags). These can be chosen freely, but should follow several best practices voted on by the community[1].

Despite these best practices, a frequent problem during automated processing of OSM public transit data is inconsistent station tagging. Problems include: (1) It is unclear whether two objects (e.g. two stop positions) belong to the same abstract station, because they are not grouped by any relation (Fig. 1, bottom). (2) Station objects are erroneously marked as members of a relation which belongs to another station. (3) Labels for the same station are highly different, contain errors (Fig. 1, top), or do not hold the station name at all (but e.g. platform numbers).

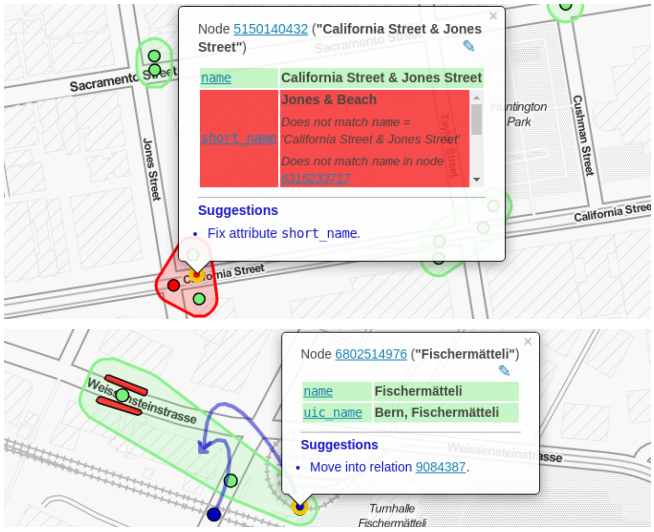


Figure 1: Top: *staty* found an erroneous `short_name` tag for a bus stop in San Francisco. "Jones & Beach" is a station several blocks north, the label was added by mistake. Bottom: staty suggesting to move two station nodes (blue) into an existing `stop_area` relation (light green) in Bern, Switzerland.

The goal of this work is to automatically find such inconsistencies and provide suggestions to map editors how to fix them.

1.1 Related Work

A large body of work exists on automated quality assessment of OSM data (see e.g. [1, 5, 6]). It may be categorized into extrinsic methods which compare OSM data to an external ground truth, and intrinsic methods operating only on the OSM data itself [1]. Extrinsic methods usually employ official datasets (e.g. [2]). These are typically hard to obtain and/or limited to a specific region. A recent work uses deep learning to compare OSM data to satellite imagery [9].

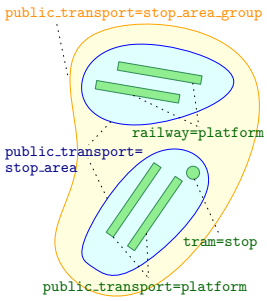
Regarding intrinsic methods, most OSM editors now come with heuristics to automatically check the quality of edits. There is also a large number of standalone tools for quality assurance[2]. However, they are typically limited to syntactic suggestions. To overcome

[1] https://wiki.openstreetmap.org/wiki/Editing_Standards_and_Conventions

[2] https://wiki.openstreetmap.org/wiki/Quality_assurance

SIGSPATIAL '20, November 3–6, 2020, Seattle, WA, USA

ACM ISBN 978-1-4503-8019-5/20/11.
https://doi.org/10.1145/3397536.3422342



lvl	tag	value
2	public_transport	stop_area_group
1	public_transport	stop_area
0	public_transport	stop_position, platform, stop, halt, station
	highway	bus_stop, platform
	railway	halt, tram_stop stop, platform
	tram	stop, platform
	subway	stop, platform

Figure 2: OSM station hierarchy used by our approach. Platforms and stop points, which can be tagged in various ways, may be grouped into a `stop_area` relation describing a single station, which can again be grouped into a `stop_area_group`.

this limitation, previous work applied machine learning to assess, correct and enrich OSM data intrinsically. For example, in [7], a random forest classifier was used to assess the quality of highway tagging. OSMRec [8] auto-suggests tags for new objects by training SVMs on existing objects and their tags. In [3] and [4], classification approaches were used to fill in missing road segments, and add points of interest (POI) tags, respectively.

The advent of route planners in the past decades has lead to a growing need of routeable geo data and an interest in assessing and improving OSM routeability in particular. For multi-modal route planners, correct station tagging is also important. While the tools mentioned above are often able to find holes in road networks or errors in turn restrictions, we are not aware of previous work which addresses the quality of public transit tagging in OSM.

1.2 Station Hierarchies in OSM

Several tagging schemata for public transit data have been used in OSM so far[3]. Regarding stations, the currently active schema Public Transport Version 2 (PTv2) is focused on describing stations through real-world physical objects and locations (timetable poles, platforms, stop positions, etc.) which are grouped into a single station entity by a `stop_area` relation. An earlier schema (PTv1) allowed these `stop_area`s to be grouped again by a super-relation `stop_area_group`. This was discouraged in 2011 by PTv2, although with over 5,000 existing relations, `stop_area_group` remains widely in use[4]. A recent proposal, called Refined Public Transport, plans to re-introduce them and generally aims to simplify PTv2. The need for automated tools to check e.g. station tags is explicitly mentioned in this proposal[5]. It is reasonable to expect that a mix of all schemata will be used for the foreseeable future.

In this work, we follow the common practice of using a three-level hierarchical approach for station tagging: Level 0 contains physical objects and locations like platforms and stop positions, but also abstract objects like label nodes at station centroids. Level 1 contains `stop_area` relations which group level 0 objects into a single station entity. Level 2 groups `stop_area` relations into `stop_area_groups` like described above. Figure 2 gives an example and lists the tags we use for each level. Note that level 0 objects cannot be direct members of a level 2 group.

[3]https://wiki.openstreetmap.org/wiki/Public_transport
[4]https://taginfo.openstreetmap.org/tags/public_transport=stop_area_group
[5]https://wiki.openstreetmap.org/wiki/Proposed_features/Refined_Public_Transport

2 PIPELINE

Our pipeline is depicted in Figure 3 and consists of 4 steps: (1) We translate the OSM stations into an abstract representation we call *station identifiers*: tuples of a label and a geographic position which are grouped into clusters. (2) We perform a pairwise similarity classification between stations within a certain distance. (3) We re-cluster the identifiers to move non-matching ones from existing clusters into new single-element clusters. Afterwards, matching clusters are merged again. (4) We derive errors and suggestions from the differences between the original and the new clustering.

2.1 Translating OSM Data to Station Identifiers

We start with OSM data filtered by the tags given in Figure 2, right. OSM offers multiple tags to label objects. For each of the kept node, way or relation objects, we use the values of the following list of tags as labels: `name`, `uic_name`, `alt_name`, `loc_name`, `nat_name`, `official_name`, `reg_name`, `ref_name`, `short_name`, `sorting_name` and `gtfs_name`. For example, in Figure 3a two stations "Park Street" and "Main Street" are described by multiple level 0 station nodes, each with different labels, and two level 1 `stop_group` relations.

If a level 0 object is part of a `stop_area` relation, we add each label of that relation as a label to each individual member (but only if the label was not already present in the member). For each level 0 object with geometry g (either a point, a polyline or a polygon) and labels $l \in L$, we create an explicit station identifier $s = (l, g)$. Two station identifiers s_1 and s_2 are *clustered* if they either belong to the same original level 0 object (e.g. if the object had multiple labels), or if their original level 0 objects were part of the same `stop_area`. In the latter case, we store the ID of the original relation for that cluster. Station identifiers which are not part of any other cluster are put into a single-element *orphan* cluster.

2.2 Similarity Classification

In a second step, we do a pairwise similarity classification between station identifiers within a threshold distance. If the distance is greater than the threshold, we implicitly assume that they are not similar. The clustering is ignored in this step. For our experiments, we used a threshold of 1,000 meters. We use a machine learning based classification approach in which a random forest classifier is trained on matching 3-grams of identifier labels, their edit distance, the geographical distance of the identifier geometry and their positions on multiple offset grids (to capture label characteristics on several regional levels). Figure 3c shows the similarity measures for two identifiers to their neighbors.

2.3 Re-Clustering

After we have obtained the pairwise identifier similarities, we perform a re-clustering. We first *remove* non-matching identifiers from their clusters. An identifier is non-matching if the similarity to half or more members of its cluster is below a threshold (0.6 in our experiments). A removed identifier is put into a new *orphan* cluster.

Afterwards we merge matching cluster again using an iterative approach. In each iteration, the pairwise similarities between clusters are determined by averaging the pairwise similarities of their members. The resulting list of merge candidates is ordered in descending manner by their similarity. We then merge the clusters in

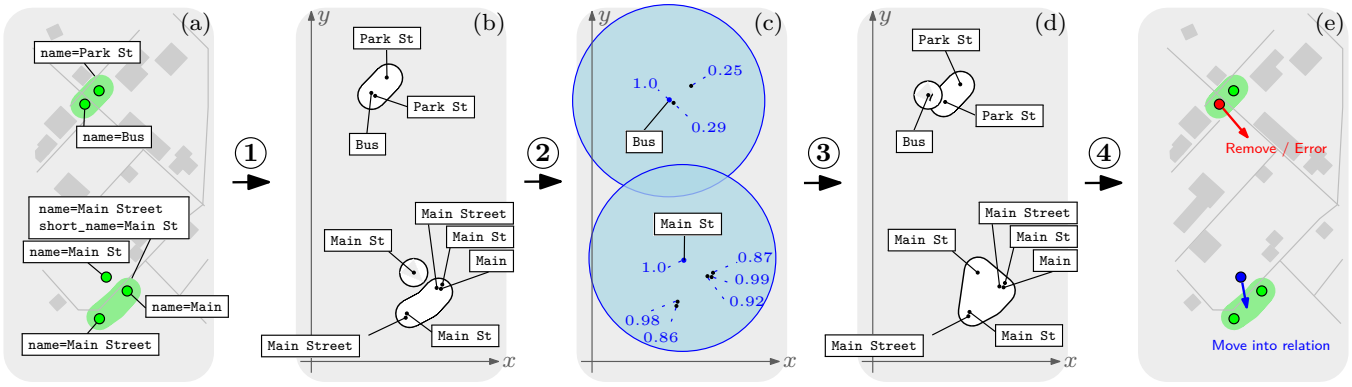


Figure 3: Our 4-step pipeline to derive errors and edit suggestions from OSM station data. The data is first transformed into clustered station identifiers (1). Afterwards, we perform a pairwise similarity classification between identifiers within a certain distance (2). The identifiers are then re-clustered based on their similarity scores (3). At the end, errors and suggestions are derived from the difference between the initial clustering and the new clustering (4).

the order in which they appear in the sorted list. If two clusters are merged, they are marked as *tainted* for the current iteration. If a tainted cluster appears a second time in this iteration, we do not merge. Figure 3d shows the re-clustered example identifiers.

2.4 Derivation of Suggestion and Errors

We then compare the new clustering to the original clustering obtained from the OSM data and derive suggestions and error messages from the differences per original level 2, level 1 or level 0 object. We distinguish the following situations:

OK All identifiers of an object are still in the same cluster.

DL The majority of identifiers of a level 0 object originally part of a `stop_area` is now in clusters not derived from a `stop_area`. We suggest to remove the object from the `stop_area`.

MV The majority of identifiers of a level 0 object originally part of a `stop_area` *A* are now in a cluster derived from a `stop_area` *B*. We suggest to move the level 0 object from *A* to *B*.

GR The majority of identifiers of a level 0 object *not* originally part of a `stop_area` are now members of a cluster derived from a `stop_area`. We suggest to move the object into the `stop_area`.

CR The majority of identifiers of a level 0 object is now a member of a cluster *not* derived from a `stop_area`, and this new cluster contains multiple level 0 objects. We suggest to create a new `stop_area` and add the objects to it.

ER None of the above apply, but an identifier is still not in its original cluster anymore. We mark the original tag the label was derived from as erroneous and list the unmatching identifiers.

MG If all members of a `stop_area` *A* were suggested to be moved into the same other `stop_area` *B*, we do not report each individual suggestion, but suggest to either *merge* the `stop_area` relations *A* and *B*, or group them into a new level 2 `stop_area_group` relation. If one of them is already part of such a `stop_area_group`, we suggest to move the other one into it.

2.4.1 Handling alt_name Tags. Special care has to be applied for labels derived from `alt_name` tags, as these alternative names usually differ greatly from other names. We do not count negative matches to station identifiers derived from `alt_name` tags when we establish the majorities described above. Positive matches, however, are counted. Similarly, in situation ER, we do not report any error if a label derived from an `alt_name` tag did not match another label.

2.4.2 Platform Names as Station Labels. We found that label tag errors (ER) were often caused because mappers incorrectly filled the `name` tag of platforms or stop positions with platform names. We catch this by a simple heuristic: if a label is marked as erroneous and is either (1) a numeric value, (2) a single letter, (3) a combination of a numeric value and a letter, e.g. "12b" or (4) a combination of a single string token with (1), (2) or (3), e.g. "Track 12b", we hint that the reason for the tag error may be because it is a platform name.

3 EXPERIMENTAL RESULTS

We implemented the approach described above in a tool called *staty*, which can be accessed under https://staty.cs.uni-freiburg.de. Our tool offers a map to browse the analyzed station data, marks errors and suggestions for all three station hierarchy levels described in Section 1.2 and offers detailed edit suggestions on click. It additionally offers a search functionality for easier navigation.

We tested both the approach and our tool on the OSM data of the United Kingdom (UK), the United States (USA) as well as on the combined dataset of Germany, Switzerland and Austria (DACH).

3.1 Distribution of Suggestions and Errors

Table 1 gives an overview over the number of errors and suggestions found by our tool for our testing datasets. For all three datasets, the most common suggestion was to group existing level 0 objects into a new level 1 `stop_area` (CR). This matches our experience with OSM data. It is common to see for example a `stop_position` and

	$\|L_0\|$	$\|L_1\|$	$\|L_2\|$	OK	DL	MV	GR	CR	ER	MG	t
USA	231k	27k	564	179k	163	57	705	75k	157	1.7k	1.6h
UK	254k	15k	163	115k	351	134	1k	150k	180	641	0.9h
DACH	704k	97k	1.2k	538k	161	335	23k	239k	1.3k	734	3.9h

Table 1: Dataset sizes, suggestion and error distribution, and analyzing time t for our three testing datasets. $|L_0|$, $|L_1|$ and $|L_2|$ give the numbers of objects on level 0, 1 and 2.

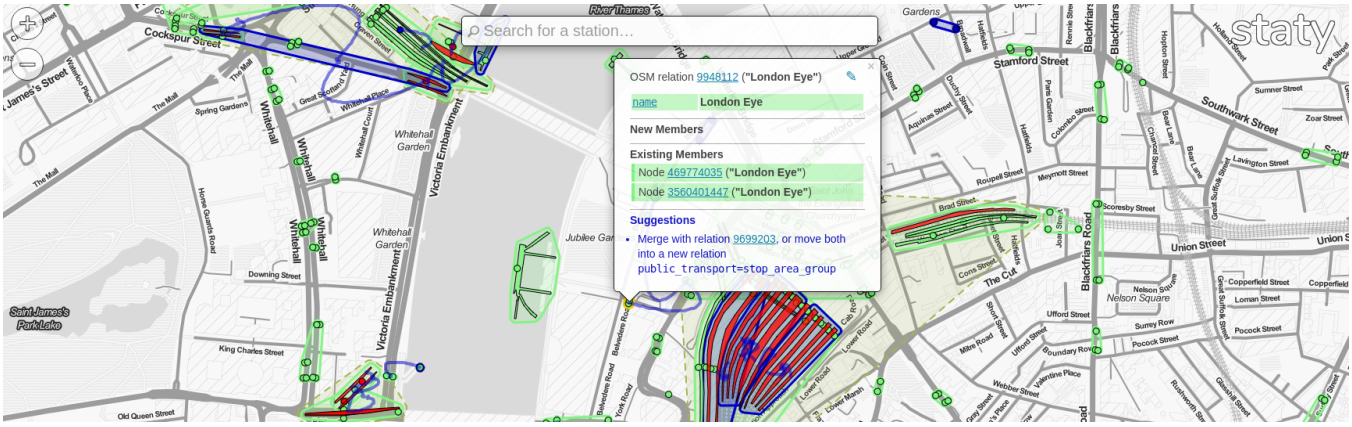


Figure 4: *staty* showing suggestions (blue) and found errors (red) in London, England. Stations for which no errors or suggestions could be found are marked green.

a `platform` object for each direction a bus stop is served, without any relation grouping them together. The most common cause for attribute errors (ER) was the incorrect usage of track numbers as station names, as was confirmed by both a manual investigation and the track number heuristic described in Section 2.4.2.

3.2 Running Time

As we want to give mappers frequent feedback, we strive for a reasonable running time of our approach. Table 1 gives the running times for our approach on our three testing datasets. Times were measured on an Intel Xeon E5649 CPU with 24 cores à 2.53 GHz. The time is given without the training time for the station similarity classifier, as this classifier does not have to be re-trained regularly. The running time was below four hours for all our datasets, which allows for multiple updates per day. Our experimental implementation updates once every 24 hours.

3.3 Correctness

As we are not aware of any fitting extrinsic ground truth dataset, we could only assess the quality of our suggestions and reported errors subjectively via random samples. From this, we found the following common mistakes made by our tool: (1) Unusual abbreviations and shortenings are not recognized by the underlying classifier, (e.g. "Hackney College" vs. "Hackney Community College"). (2) In cities with a regular road grid, stations describing different locations at the same intersection like "41st St & 8th Av" and "8th Av & 41st St" are incorrectly suggested to be grouped into a level 1 relation. (3) Unusually large geographical distances between identifiers lead to attribute errors (ER) being reported.

4 CONCLUSIONS AND FUTURE WORK

We described an approach for automated quality assurance of public transit station tagging in OSM. We also presented staty, a full implementation of this method in a browser-based tool. First experimental results on the datasets of the United States, the United Kingdom, and on a combined dataset of Germany, Austria, and Switzerland already look promising and show that our method is indeed able to make valuable edit suggestions. Our experiments also show that (given an already trained classifier) the running time of our approach is still manageable (under 4 hours) even for entire countries, enabling regular full updates. Nevertheless, it would be interesting to incorporate minutely incremental updates.

A thorough investigation of the quality of our suggestions and reported errors would require an extended user study with professional members of the OSM mapping community. To assess the quality of the station similarity classifier that lies at the core of our method, a comprehensive evaluation against ground truth data would be a valuable contribution.

Further, it would be a valuable addition if staty also made suggestions for non-label tags used in public transit stations. For example, it would be easy to add static linting to tags which have a restricted set of values or to check for the presence of suggested tags in particular station objects. If would also be interesting to check `stop_group` relations for the presence of exactly one main object which aggregates the station's meta tags, as it is proposed in the upcoming Refined Public Transport schema. Our tool could also suggest such a main object automatically. This would make staty a valuable contribution to the acceptance of the new schema.

REFERENCES

[1] Christopher Barron, Pascal Neis, and Alexander Zipf. 2014. A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis. *Transactions in GIS* 18, 6 (2014), 877–895.
[2] Hongchao Fan, Alexander Zipf, Qing Fu, and Pascal Neis. 2014. Quality assessment for building footprints data on OpenStreetMap. *International Journal of Geographical Information Science* 28, 4 (2014), 700–719.
[3] Stefan Funke, Robin Schirrmeister, and Sabine Storandt. 2015. Automatic Extrapolation of Missing Road Network Data in OpenStreetMap. In *2nd International Workshop on Mining Urban Data, Lille, France*. 27–35.
[4] Stefan Funke and Sabine Storandt. 2017. Automatic Tag Enrichment for Points-of-Interest in Open Street Map. In *Proceedings of the 15th W2GIS 2017, Shanghai, China*. 3–18.
[5] Jean-François Girres and Guillaume Touya. 2010. Quality Assessment of the French OpenStreetMap Dataset. *Transactions in GIS* 14, 4 (2010), 435–459.
[6] Mordechai Haklay. 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 (2010), 682–703.
[7] Musfira Jilani, Padraig Corcoran, and Michela Bertolotto. 2014. Automated highway tag assessment of OpenStreetMap road networks. In *Proceedings of the 22nd ACM SIGSPATIAL, Dallas/Fort Worth, USA*. ACM, 449–452.
[8] Nikos Karagiannakis, Giorgos Giannopoulos, Dimitrios Skoutas, and Spiros Athanasiou. 2015. OSMRec Tool for Automatic Recommendation of Categories on Spatial Entities in OpenStreetMap. In *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria*. ACM, 337–338.
[9] Xuejing Xie, Yi Zhou, Yongyang Xu, Yunbing Hu, and Chunling Wu. 2019. OpenStreetMap Data Quality Assessment via Deep Learning and Remote Sensing Imagery. *IEEE Access* 7 (2019).