Using Metamorphic Testing to Improve the Quality of Tags in OpenStreetMap

Jesús M. Almendros-Jiménez, Antonio Becerra-Terón, Mercedes G. Merayo, and Manuel Núñez

Abstract—We present a metamorphic testing approach to validate the information included in OpenStreetMap, a collaborative effort to produce a free map of the world. We focus on the quality of the tags storing the information about the elements of the map. We identified metamorphic relations with the potential to detect different types of tagging errors. In particular, we carefully designed mechanisms to automatically generate *follow-up inputs*, a fundamental component in the successful application of a metamorphic testing approach. The intrinsic nature of automatically analysing tags implies that we will detect real errors but some false positives as well. In order to obtain a good trade-off between real errors and false positives, we introduce *thresholds*. Our MRs will raise an error associated with a certain value if, depending on the nature of the MR, we have a certain number of elements (not) fulfilling a given condition. In order to evaluate the goodness and versatility of our framework, we chose four cities in different continents with the goal of analysing very heterogeneous contributors adding information in different languages. The application of this framework to the analysis of the chosen cities revealed errors in all of them and in all the considered categories. In addition, around 66% of the errors found by our MRs in the analysed areas have not been previously reported by *Osmose*, the de facto standard OSM error checker.

Index Terms—Metamorphic testing, OpenStreetMap, quality of maps

1 Introduction

Software Testing [1] is the main Software Engineering validation technique. Usually, in testing we apply inputs to the System Under Test (SUT) and decide whether the observed outputs are the expected ones. In order to make a verdict about the correctness of the observed outputs, we rely on an *oracle*. *Metamorphic Testing* [2], [3], [4] (MT) has been successfully used to test systems when an oracle is not available [5]. MT depends on the definition of *Metamorphic Relations* (MRs) between the inputs that we apply to the SUT and the observed outputs. If we find a violation of an MR, then we know that the system is not correct. For example, consider the sine function, a classical and simple example to show the characteristics of MT. Let P be a (potentially wrong) implementation. Even if we do not know the result that P has to produce for a certain value x, we do know that P must fulfil some properties. For instance, if we compute $P(x)$ and $P(x+2\cdot\pi)$ and we obtain different results, then we know that P is faulty. A formal definition of this MR is as follows. Let $i_1 \in \mathbb{R}$ and $i_2 = i_1 + 2\cdot\pi$:

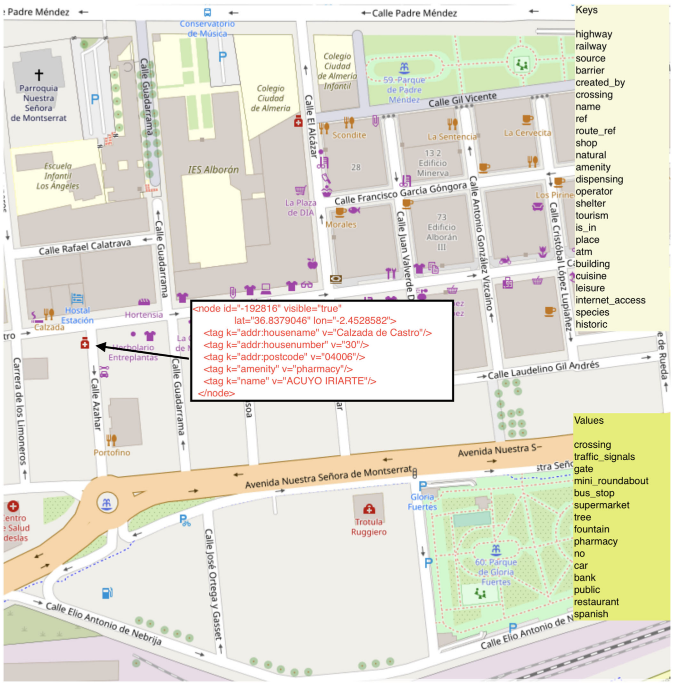
$$\begin{gathered} MR_{+2\cdot\pi}(i_1, i_2, P(i_1), P(i_2)) \\ \Updownarrow \\ P(i_1) = P(i_2) \end{gathered} \tag{1}$$

Quoting from [6]: "When implementing MT, some program inputs (called *source* inputs) are first generated as source test cases, on the basis of which an MR can then be used to generate new inputs as *follow-up* test cases." Usually, source inputs are provided by the tester, using its expertise concerning the expected behaviour of the SUT. Besides, follow-up inputs should be (as much as possible) automatically generated from the corresponding source inputs. In the approach reported in this paper, all the follow-up inputs are automatically generated. In addition, with the exception of one MR, we are able to automatically generate source inputs from an initial *seed*, the corresponding follow-up inputs and the observed outputs. Let us consider again the sine function. A tester may consider that 0 is an interesting value to check the SUT. This will be a source input. Using the definition of the MR and this source input, we automatically obtain the follow-up input $2\cdot\pi$ and we should check $MR_{+2\cdot\pi}(0, 2\cdot\pi, P(0), P(2\cdot\pi))$, that is, whether $P(0) = P(2\cdot\pi)$. Now, this process can be (automatically) iterated. We may consider $2\cdot\pi$ as source input, *generate* the follow-up input $4\cdot\pi$ and check whether $P(2\cdot\pi) = P(4\cdot\pi)$.

In this paper we apply MT to find errors in OpenStreetMap (OSM) [7], [8], a collaborative and open source initiative that is

- *Jesús M. Almendros-Jiménez and Antonio Becerra-Terón are with the Department of Informatics, Carretera de Sacramento s/n, University of Almería, 04120 Almería, Spain. E-mail: {jalmen, abecerra}@ual.es.*
- *Mercedes G. Merayo and Manuel Núñez are with the Institute of Knowledge Technology, Universidad Complutense de Madrid, 28040 Madrid, Spain. E-mail: mgmerayo@fdi.ucm.es, mn@sip.ucm.es.*

Manuscript received 23 September 2021; revised 25 February 2022; accepted 28 February 2022. Date of publication 7 March 2022; date of current version 13 February 2023.
This work was supported in part by the State Research Agency (AEI) of the Spanish Ministry of Science and Innovation under Grants PID2019-104735RB-C42 (SAFER) and RTI2018-093608-B-C31 (FAME), in part by Comunidad de Madrid under Grant S2018/TCS-4314 (FORTE-CM) co-funded by EIE Funds of the European Union.
(Corresponding author: Manuel Núñez.)
Recommended for acceptance by D. Hao.
Digital Object Identifier no. 10.1109/TSE.2022.3156182



```
<node id="-192816" visible="true"
          lat="36.8379046" lon="-2.4528582">
   <tag k="addr:housename" v="Calzada de Castro"/>
   <tag k="addr:housenumber" v="30"/>
   <tag k="addr:postcode" v="04006"/>
   <tag k="amenity" v="pharmacy"/>
   <tag k="name" v="ACUYO IRIARTE"/>
 </node>
```

Fig. 1. A graphical OSM of Almería (excerpt).

currently used by many companies and organisations around the world[1]. Our approach can be used to assess the quality of software systems which work or depend on OpenStreetMap datasets. The typical example is the OSM API[2] whose correct behaviour can be only tested when datasets are valid. However, this is not the only case. Well-known rendering systems, such as Osmosis, OSM2PGSQL, and Mapnik, among others, are frequently used by developers to import and visualize OSM datasets. In order to properly work, these systems need validated maps. In fact, a considerable amount of queries on the OSM ticket system are referred to rendering systems and their correct behaviour, whose failures are frequently due to OSM data quality issues.

Volunteers contribute to OSM, with the goal of creating an editable and free map of the world, with spatial data (i.e., geometries) and textual data in the form of *tags*. Geometries are used to denote *points*, called nodes in OSM, such as trees and traffic signals, and *polylines*, called ways in OSM, representing structures such as highways and buildings. The application of MT to the analysis of geometries was the goal of our recent work [9]. In this paper we are interested in tags. A tag is simply a (key, value) pair, where the value is assigned to a key that denotes a certain characteristic. Tags provide information about the elements of the map, in particular about points of interest such as museums, hotels, touristic places, etc. Fig. 1 presents a portion of the map of Almería, Spain. The description of a node (see boxed item) includes its unique identifier and coordinates, together with a set of tags: (key, value) pairs. Tags indicate the main characteristics of the element, including its *name*. In this case, we have a pharmacy (indicated by the tag (amenity, pharmacy)), located at the *Calzada de Castro* street, number 30, and whose name is *ACUYO IRIARTE*.

OSM contributors should follow some rules[3] in order to facilitate how OSM tools and users interpret the information included in a map. OSM elements can include an unlimited number of tags describing each feature. Although (key, value) pairs can be freely selected, some recommendations are informally proposed. In particular, some fixed (key, value) pairs, or even fixed combinations of keys, should be used to describe the same kind of structure, some of them are mandatory and others are recommended. Information about the so-called *folksonomy* [10], that is, the tags used to thematically describe OSM elements, has been collected and aggregated by several websites such as *Taginfo*[4], which uses a full OSM database to report its statistics, and *Tagfinder*[5].

The main goal of our approach is to reveal tagging mistakes in OSM without requiring external resources, that is, without using an oracle. The main motivation behind this work is to increase the quality of maps by, as much as possible, automatically analysing them. Our previous experience taught us that finding *single* errors is very difficult and costly: an expert must review, by hand, the tags of the map. This usually requires to manually check conformance with tagging conventions. Typically, the detection of tagging mistakes is carried out by the community of contributors itself, in which contributors review each other.

The aim of this proposal is to detect *agreement patterns:* if something is repeated many times, then a deviation from it could mean that a contributor does not follow the (implicit) tagging standards. In other words, the disagreement of contributors in a certain map will reveal tagging mistakes. The objective is that if some OSM feature deviates from the tagging norm, then the MRs will consider the tagging of the OSM feature as an error. However, this information could be valid according to the OSM rules point of view. In this case, we have a false positive. Let us remark that most OSM contributors are not aware of all the recommendations of OSM. The knowledge of all the keywords is already difficult and the combination with values and other keys is still harder. The number of rules is huge and most of them are expressed by semantic concepts related to topological and cartographic aspects. This motivates the design and development of systems aiming to assist the recommendation of tags and the detection of tagging problems. However such systems have to deal with the following problem. The labelling of an object depends on the type and characteristics of the object and the contributor is the only one that knows which type of object wants to label. Thus, the identification of tagging errors is hard. Moreover, errors are of different nature. Let us suppose that an unknown (i.e., not allowed) label is used. This type of error can be easily detected, even though the number of allowed OSM labels is huge. Let us suppose now that a known (i.e., allowed) label is incorrectly used (i.e., wrongly combined with others). This is a more serious problem because the detection requires to know what type of object the contributor wants and, even more

1. https://wiki.openstreetmap.org/wiki/They_are_using_OpenStreetMap
2. https://wiki.openstreetmap.org/wiki/API
3. https://wiki.openstreetmap.org/wiki/Editing_Standards_and_Conventions
4. http://taginfo.openstreetmap.org
5. http://tagfinder.herokuapp.com

difficult, the OSM rules to be applied. A suitable strategy can be to report suspicious labels, which are labels that seem to be out of context. But even in this case, some labels are rarely used but allowed, which provokes false positives.

Any characterisation of a contributors' agreement will require a significant number of occurrences and a low number of discrepancies. In order to properly include *quantification* into MRs, we will use thresholds in their definition so that the testers can formally quantify when they consider to have many/few occurrences of a certain phenomenon. The performed experiments have shown that some combinations of thresholds, depending on the map size and the applied MR, are more likely to find real errors and reduce the number of false positives. These values can be used by other testers when analysing their maps. We are aware that by using thresholds, we will overlook some real errors and some users of the approach might prefer not to use them.[6] However, our preliminary experiments detected a vast amount of false positives if we do not use thresholds.

In this paper we present a set of MRs that can be used to analyse the tags associated with elements of OSM. The MRs code is freely available. The MRs can be used by testers in two different ways. First, they can be used to analyse the same properties that we present in a different map. In this case, testers do not need to implement anything. Second, if testers need to extend our framework with other MRs, then they can follow the pattern given by the available MRs. Note that the implementation of new MRs is simplified because we use a declarative programming language whose syntax is close to their mathematical formulation. In order to apply MRs, testers need to provide appropriate source inputs but the rest of the operative is almost automatic.

In order to assess the usefulness of the proposed MT approach, we have applied our MRs to evaluate the tags labelling elements of four different cities. All the MRs found potential errors in all the cities: a total of 496 were actual errors. Moreover, we checked whether these errors had been previously reported in the most used OSM error checker, *Osmose*[7]: only 184 errors of them are currently reported by *Osmose*. Therefore, approximately 63% of the errors found by our approach have not been (yet) reported by *Osmose*.

The rest of the paper is structured as follows. In Section II we present related work. In Section III we define some notions that will be used along the paper. Section IV presents the set of MRs and some implementation details. In Section V we present our research questions and the experiments that we performed to answer them. Finally, in Section VI we present the conclusions and discuss some lines for future work.

2 Related Work

In this section we will review related work on the areas on metamorphic testing and on analysis of OSM.

MT is a mature field and there are several complete surveys outlining the main contributions, lines of work and challenges [6], [11]. MT has been applied to very heterogeneous fields such as citation database systems [12], class integration test order generation [13], cloud systems [14], compilers [15], cybersecurity [16], post-quantum cryptography [17], program repair [18], RESTful web APIs [19], search engines [20] and simulation [21], among others. More related to our application of MT to OSM, we can mention the use of MT in data validation [22]. This work is related to ours because the main goal of our framework is to find *errors* in the data of a map. The notion of *metamorphic exploration* [23] is also relevant for our work. There are situations where an MR reports a potentially incorrect behaviour, which turns out to be correct after further exploration. This may help to better understand a certain behaviour of the SUT. Our thresholds play, using a different mechanism, a similar role: if something happens more/less times than expected (that is, more/less times than the threshold allows), then we might have detected some real errors. In addition, if they were false positives, they might indicate a non appropriate, although allowed, use of the tagging conventions. Thresholds have been used to define an MR to test MISA-RT, a simulation program that computes the density of material defects [24]. Instead of testing with different thresholds, the threshold is set to the optimum value of a function relating inputs and outputs.

The application of MT to analyse maps has been very limited. We have recently used it [9] to find errors in OSM concerning the representation of spatial data. We were able to find different categories of errors such as ways without entrances or exits, deadlocked ways, unconnected roundabouts, overlapping between buildings, etc. However, the type of errors that we were targeting was completely different (catching geometry errors was simpler than catching inconsistencies in the data) and thresholds, a novel feature, bring an additional complexity to the implementation, validation and application of the new framework. A previous approach used MT to validate maps [25], [26], but the focus was on analysing distances and paths while, as we have already explained, we are interested in the information included in the map. MT has been also applied to Geographic Information Systems [27], where the main focus was on evaluating the calculation of superficial areas. The MRs were defined on the basis of different artefacts (requirements, program properties, algorithms, etc) and applied to a case study, based on mutation testing, to assess the usefulness of the approach. Therefore, this approach and ours are unrelated.

There exist several methods [28], [29], [30], [31] to assess the inappropriate use of tags in OSM maps. In addition, the community has developed many tools to support this task[8] and we can mention *iOSMAnalyzer* tool[9], *Quantum GIS (QGIS)*[10], *OSMantic*[11], *Keep Right*[12] and *OSM inspector*[13]. The most widely used is *Osmose (Open Street Map Oversight Search Engine)*.

Some of the previous methods carry out an *extrinsic quality analysis* by using an *authoritative dataset* that serves as a *ground truth dataset*. Therefore, they rely on a type of oracle

6. It is very simple to discard thresholds: it is enough to set them to either 0 or ∞, depending on whether we are asking for a certain value to be greater, respectively smaller, than the threshold.
7. http://osmose.openstreetmap.fr/es/map/

8. https://wiki.openstreetmap.org/wiki/Quality_assurance
9. https://github.com/zehpunktbarron/iOSMAnalyzer
10. https://www.qgis.org/
11. https://sites.google.com/site/osmanticjosm/
12. https://keepright.at/
13. https://tools.geofabrik.de/osmi/

and, as such, are not related to our approach. In contrast, *intrinsic quality analysis* validates the information included in OSM by assuming some rules of quality [31], [32], [33], [34], [35], [36], [37], [38]. The proposed MT approach can be considered an intrinsic quality analysis in which MRs are used to assess the quality of OSM maps.

We have previously worked on OSM and quality assessment [39], [40], [41]. We defined an extrinsic quality method to evaluate the tags labelling Spain OSM maps by comparing them with the Taginfo[14] dataset. Therefore, the work reported in this paper presents a different approach, intrinsic versus extrinsic, using a completely different underlying technique (we strongly rely on MT and MT was not used in the aforesaid work), and supported by a fully implemented and highly automatic framework.

3 Preliminaries

In this section we review the main concepts and notions used in our framework. We use a slightly modified version of the definitions introduced in our previous work [9], so that we can focus on tags information.

Definition 1. *We distinguish three categories of OSM data types: tags, nodes and ways.*

- *We consider that a* tag *is given by a pair of strings (k, v), where k represents a key while v represents a value. Tags will provide information about nodes and ways.*
- *We have that a* node *is given by a triple $n = (id, (lat, long), Tg)$, where id is a string that unequivocally identifies the node, lat and $long$ are the latitude and longitude of the node in the World Geodetic System (WGS84) and Tg is a set of tags providing information about the characteristics of the node.*
- *Finally, a* way *is a triple $w = (id, \langle n_1, \ldots, n_s\rangle, Tg)$ where id and Tg play the same role as in nodes and $\langle n_1, \ldots, n_s\rangle$ is the sequence of nodes that conform the way.*

Let $\mathcal{N}$ be a set of nodes. We consider that $W_{\mathcal{N}}$ represents the set of ways such that all the nodes appearing in a way belong to $\mathcal{N}$. Given a node or way e, $tags(e)$, $keys(e)$ and $vals(e)$ represent, respectively, the tags of e and the keys and values appearing in any tag of e. Given a set of elements E, we define $tags(E) = \bigcup_{e\in E} tags(e)$, $keys(E) = \bigcup_{e\in E} keys(e)$ and $vals(E) = \bigcup_{e\in E} vals(e)$. Finally, $\overline{E}$ denotes the complement of set E.

Intuitively, an OSM map is a set of nodes $\mathcal{N}$ and a set of ways $W_{\mathcal{N}}$ constructed from the nodes belonging to $\mathcal{N}$. Each way $w \in W_{\mathcal{N}}$ includes a unique identifier, a set of tags and the non-empty set of nodes that constitute the way.

Definition 2. *Given a fixed set of nodes $\mathcal{N}$, we say that an* OSM map *is a pair $m = (\mathcal{N}, \mathcal{W}_{\mathcal{N}})$. Abusing the notation, we will write $e \in m$ if $e \in \mathcal{N} \cup \mathcal{W}_{\mathcal{N}}$.*

We force OSM maps to fulfil a usual restriction: for each $e \in m$ and $k \in keys(e)$, there exists a unique value v such that $(k, v) \in tags(e)$.

Instead of using a different name for each map, we will denote by $\mathcal{M}$ the specific map that we are considering at a certain moment.

The basic idea underlying an MT approach is that we apply inputs to the system under study, record the produced outputs, and check whether applied inputs and observed outputs violate some relations. Since we are working within a *non-standard* type of system, it is necessary to formally define these concepts. First, we define inputs and outputs.

Definition 3. *The set of* inputs*, denoted by $\mathcal{I}$, is given by the minimum set of expressions that can be derived from the non-terminal symbol I of the following BNF:*

$$\begin{aligned} I &:= (T, C)\\ T &:= * \mid node \mid way\\ C &:= \{L\}\\ L &:= (string, string) \mid (string, string), L \end{aligned}$$

The application of an input $i = (T, C)$ to a map $\mathcal{M}$ produces as output*, denoted by $\mathcal{M}(i)$, the elements of the map of type T such that they fulfil the conditions included in C. If C is a singleton (in many practical situations, this will be the case), then we simply write (k, v) instead of $\{(k, v)\}$.*

Intuitively, an input $i = (T, C)$ denotes that we are looking for elements of a map of type T (nodes, ways or both) that must fulfil the set of conditions included in C. L is the list of elements of this set, so that C is simply the set of elements appearing in L. Each element of L is defined by a pair of strings: a key and a value. As a special case of *string*, we consider the wildcard character $*$. As usual, $*$ matches any string of zero or more characters. For the sake of simplicity, we do not distinguish between capital and lowercase letters in strings. Next, we present a simple example.

Example 1. A condition such as $(name, hotel*)$ will be matched by those elements having a $(name, v)$ tag such that v matches the pattern $hotel*$ (e.g. $Hotel\ California$). A condition such as $\{(name, A7), (highway, *)\}$ will be matched by elements being part of a highway whose name is $A7$. Technically, we will select those elements e such that there exist $(name, v_1), (highway, v_2) \in tags(e)$, with $v_1 = A7$.

Finally, we present our notion of MR, which is a bit different from conventional ones due to the use of *thresholds*. As usual, MRs relate inputs and their corresponding outputs. In addition, and this is a non-standard feature that in particular was not used in our previous work [9], our MRs have thresholds. These values will be used to impose conditions on the number of elements that must fulfil a certain property taking part in the definition of an MR.

Definition 4. *Given an OSM map $\mathcal{M} = (\mathcal{N}, \mathcal{W}_{\mathcal{N}})$ and $n \geq 0$ thresholds $\delta_1, \ldots, \delta_n \in \mathbb{N}$, a* metamorphic relation *(MR) $R \subseteq \mathcal{I}^2 \times \mathcal{P}(\mathcal{N} \cup \mathcal{W}_{\mathcal{N}})^2 \times \mathbb{N}^n$ is a relation over 2 inputs, i_1, i_2, their corresponding outputs, $\mathcal{M}(i_1), \mathcal{M}(i_2)$, and n thresholds. We write $R(i_1, i_2, \mathcal{M}(i_1), \mathcal{M}(i_2), \delta_1, \ldots, \delta_n)$ to denote $(i_1, i_2, \mathcal{M}(i_1), \mathcal{M}(i_2), \delta_1, \ldots, \delta_n) \in R$.*

Given n thresholds $\delta_1, \ldots, \delta_n$, inputs $i_1 = (T_1, C_1), i_2 = (T_2, C_2)$ and their corresponding outputs $O_1 = \mathcal{M}(i_1), O_2 =$

14. https://taginfo.openstreetmap.org/

$\mathcal{M}(i_2)$, in order to define an MR R, we use the following pattern.

$$R(i_1, i_2, O_1, O_2, \delta_1, \ldots, \delta_n)$$
$$\Updownarrow_{\mathbf{def}}$$
$$\mathcal{C}(\delta_1, \ldots, \delta_n, T_1, C_1, T_2, C_2, O_1, O_2)$$

where $\mathcal{C}$ is a formula in first order logic without free variables.

Let $R(i_1, i_2, O_1, O_2, \delta_1, \ldots, \delta_n)$ be an MR. We say that i_1 is a source input *and i_2 is a* follow-up input.

In general, source inputs are provided by testers while follow-up inputs should be (as much as possible) automatically generated from source inputs and the definition of the MR. We would like to give an explanation of the important role played by thresholds in our framework.

The OpenStreetMap community provides guidelines and recommendations about how OpenStreetMap features have to be labelled. Such guidelines and recommendations are available in the OpenStreetMap Web site, organized by categories[15]. Although contributors should follow the guidelines, due to the crowdsourcing nature of OpenStreetMap, any registered contributor can violate such rules by lack of knowledge or misunderstanding, but also due to multiplicity of similar types and vague boundaries between them [42] and as an act of vandalism [43]. Definitions of some of the tags in the OSM Web site are unclear and some definitions may apply to several features.

Although any registered contributor can update the information about an OSM feature, the majority of updates are done by experts (users with a high number of contributions) and groups of users that act in a coordinated way in some areas following the same guidelines. In a crowdsourcing context, if a high number of contributions agree in a certain labelling, then it is possibly correct, while few ones can be considered as labelling errors or, at least, suspect labelling. In other words, we consider the following principle: what the majority does is presumably correct and all of the exceptions are suspicious. This principle has a usability basis: heterogeneity facilitates the interpretation of OSM features by users and tools. Previous work [32] suggested the analysis of the number of contributors who have worked on a given feature for giving more confidence. The assumption that as the number of contributors increases so does the quality is known within the open source community as "Linus' Law". Here, instead, we assume that agreement between contributors in the labelling of the same type of feature increases its reliability. The problem now is what is a *high* or *low* number of contributions. For this reason, our metamorphic rules use thresholds. They allow users to customise the MR to be more or less restrictive. A *primary* threshold considers the number of agreements in contributions, according to the above principle. Additionally, our MRs have a second threshold which serves, in combination with the previous one, to measure few contradictory occurrences. Threshold values should vary according to the frequency of a feature and the size of the studied map. In general, a given labelling is considered correct when is frequent and is considered suspicious when is scarce and there are numerous contradictory cases.

15. https://wiki.openstreetmap.org/wiki/Map_features

We are aware that by using thresholds we will not find some real errors. However, our preliminary experiments detected a vast amount of false positives if we do not use them (i.e. if we set them to 0 or ∞, depending on whether the threshold is a lower or an upper bound). A typical situation that will escape to our analysis appears when a malicious/inexpert user has a high number of contributions that systematically violate the same OSM rule. It could happen in bulk imports, which are occasionally undertaken as a supplement to data collected by individual contributors. A typical source of false positives, which again will not be detected, appears with correct OSM rules scarcely used due to lack of knowledge, existence of several rules for the same case, recommended but not mandatory rules, low labelling activity in the area, etc. Finally, note that in our previous work [9] we did not deal with the problem of false positives because we were detecting errors in the geometries of a map. For example, if a roundabout has no exit or entry, then we are sure that we have found an error.

4 Formal Definition of Our MRs

This section introduces our MRs. They are applied to inputs and outputs of a specific map and aim to establish relations that the elements of the map satisfying certain conditions must fulfil. Specifically, our MRs evaluate whether the tags labelling the elements of our map include information that is most likely wrong.

4.1 `TagsCOMPkvk`: Compatibility Between (key,value) and Key in the Same Element

The goal of our first MR is to detect elements with a certain tag such that the nature of the key and value makes it incompatible with tags having a certain key. Next, we give a simple example to motivate the usefulness of this MR. Consider a node having as one of its tags $(highway, traffic\_signals)$. Good practices in OSM say that traffic signals should not have a name. Therefore, this node should not have the tag $(name, \beta)$, for a certain string β. As we have previously commented when introducing thresholds, we might relax the last constraint and consider that a few *divergences* are allowed.

Given two thresholds $\delta, \epsilon \in \mathbb{N}$ and two strings α and β, associated with a key and a value, respectively, let $i_1 = (*, (\alpha, \beta))$ and $O_1 = \mathcal{M}(i_1)$. Given $\alpha' \in keys(O_1) \setminus \{\alpha\}$, let $i_2 = (*, (\alpha,' *))$ and $O_2 = \mathcal{M}(i_2)$.

$$\texttt{TagsCOMPkvk}(i_1, i_2, O_1, O_2, \delta, \epsilon)$$
$$\Updownarrow_{\mathbf{def}}$$
$$|O_1| \geq \delta \Longrightarrow |O_1 \cap O_2| \geq \epsilon$$

Intuitively, if there is a big enough number of (α, β) tags, then this MR will check that the number of occurrences of α' in those elements is not small.

Example 2. Consider the OSM map given in Fig. 2 where we have removed data irrelevant for the forthcoming examples (e.g. latitude and longitude of the nodes have been removed). Let $\alpha = railway$ and $\beta = stop$. Therefore, $i_1 = (*, (railway, stop))$. Then, $\alpha' \neq railway$ must be set to a key that appear in a tag of the elements having $(railway, stop)$. For example, let $\alpha' = subway$ and,

```
<node id="1">
    <tag k="public_transport" v="stop_position"/>
    <tag k="railway" v="stop"/>
    <tag k="subway" v="yes"/>
</node>
<node id="2">
    <tag k="public_transport" v="stop_position"/>
    <tag k="railway" v="stop"/>
    <tag k="subway" v="yes"/>
</node>
<node id="3">
    <tag k="public_transport" v="stop_position"/>
    <tag k="railway" v="stop"/>
    <tag k="wrong1" v="yes"/>
    <tag k="train" v="no"/>
</node>
<node id="4">
    <tag k="public_transport" v="stop_position"/>
    <tag k="railway" v="stop"/>
    <tag k="train" v="no"/>
</node>
<node id="5">
    <tag k="wrong2" v="stop"/>
    <tag k="railway" v="stop"/>
    <tag k="train" v="no"/>
</node>
<node id="6">
    <tag k="name" v="Railway Train Station"/>
</node>
```

Fig. 2. A textual and simplified OSM map.

therefore, $i_2 = (*, (subway, *))$. If we apply the corresponding inputs to the map, we obtain two outputs. For example, O_1 includes all the features having a tag $(railway, stop)$. Next, we have to check TagsCOMPkvk$(i_1, i_2, O_1, O_2, \delta, \epsilon)$. First, $|O_1|$ is equal to the number of features in the map having the tag $(railway, stop)$. For example, if we set $\delta = 6$ then $|O_1| \geq 6$ does not hold (actually, $|O_1| = 5$) and we do not check the right hand side of the implication (therefore, no errors are detected). If we set $\delta = 5$, then we have to check whether the number of elements having the tag $(railway, stop)$ and such that another of its tags has $subway$ as key is greater that ϵ. For example, if we set a small value of ϵ, for example $\epsilon = 2$, then we do not find any potential error. If we use an intermediate value such as $\epsilon = 3$, then we find two potential errors, which are, in this case, false positives. In case $\alpha' = wrong1$, $\delta = 5$ and $\epsilon = 2$ are selected, one potential error is found corresponding with a real error.

Concerning the thresholds, δ should be selected according to the frequency of the (α, β) tag. δ is used to fix the minimum number of elements in the map having (α, β) as one of its tags that are needed to start the testing process. If δ is big then the MR is only applied to relevant features of the map (whatever the size of the map is). In the experiments reported in this paper, we have set δ to 10. For example, we will have more than 10 elements when using tags such as $(building, yes)$ or $(highway, residential)$. However, using $\delta = 10$ for the $tourism$ key is not useful because of the low number of elements having tourism as key (and thus, given β, few occurrences of $(tourism, \beta)$). For this reason, for these cases we had to set δ to either 6 or 4. Naturally, the key factor for selecting 10 or a lower value is the total number of elements fulfilling the seed. However, it could happen that some interesting cases are not analysed if $\delta = 10$. For instance, the combinations of $building$ and $highway$ with values having less than 10 occurrences. Note that using a small value of δ when the number of elements having the tag given by the seed is large will reveal more errors but the number of false positives will increase as well. With respect to ϵ, its purpose is to fix the number of coincidences of (α, β) and α'. We have set $\epsilon = 2$ for all the experiments. Obviously, if the number of elements having (α, β) as tag is low (that is, we are using a small δ), then it is more likely to find false positives for α'. Finally, note that if we have few elements fulfilling the seed, then it is unlikely that we will be able to detect a wrong α'.

In order to effectively apply this MR, we need to produce (source and follow-up) inputs, apply them to the map under study, and obtain the corresponding outputs. First, we have to identify an α_0 key and a β_0 value and let $i_0 = (*, (\alpha_0, \beta_0))$ be the source input. Although this *seed* can be (manually) chosen by the tester, it will be a good practice to choose a key that appears in a big proportion of tags, so that we (recursively) cover a big portion of the map. Next, we show how follow-up inputs can be automatically generated. First, we compute the set $O_0 = \mathcal{M}(i_0)$. If this set has less than δ elements, then we stop the process because the right hand side of the MR will not be invoked. Let $A_0 = \{\alpha \in keys(O_0) | \alpha_0 \neq \alpha\} = \{\alpha_0^1, \ldots, \alpha_0^{|A_0|}\}$. If this set is empty, then we also stop the process. For each $1 \leq s \leq |A_0|$, let $i_0^s = (*, (\alpha_0^s, *))$ and $O_0^s = \mathcal{M}(i_0^s)$. We sequentially check the following $|A_0|$ metamorphic relations:

$$\texttt{TagsCOMPkvk}(i_0, i_0^s, O_0, O_0^s, \delta, \epsilon)$$

where $1 \leq s \leq |A_0|$. If any of these MRs does not hold, then we have found a failure and could stop.[16] Otherwise, we can iterate the previous process by considering each of the values $\alpha \in A_0$ as the key component of a new source input and a value β such that more than δ elements of the map include the (α, β) tag.

In order to increase automation, we provide an alternative implementation of this MR such that the tester only needs to provide the value of α_0. Our framework automatically computes $\beta_1, \ldots, \beta_\lambda$ such that each of the pairs $(\alpha_0, \beta_1), \ldots, (\alpha_0, \beta_\lambda)$ is tagging at least δ elements of the map under study and applies the previous process for the source inputs $(*, (\alpha_0, \beta_r))$, with $1 \leq r \leq \lambda$.

During the definition of our framework, we considered other MRs that ended up being particular cases of this one. It is worth to mention TagsCOMPkk, whose goal was to check that the map under study did not include tags with incompatible keys. This is the case, for instance, of *building* and *highway*. Therefore, if two keys simultaneously label a *small* number of elements of the map, then it is very likely that they should not be part of tags of the same element. Following this *compatibility* pattern, we also considered, and later discarded because they could also be simulated by using TagsCOMPkvk, the following two MRs:

- TypeCOMPk: Compatibility between the type of element (node or way) and keys included in its tags. There exist certain keys that should not appear in a tag of certain elements. For example, if we are analysing the information included in a *node*, then we

16. In practice, we do not stop because we continue traversing the map to find more errors.

should not expect that this element has a tag including *area* as its key.

- TypeCOMPv: Compatibility between the type of element (node or way) and values included in its tags. For example, if an element is a *way*, then no tag should have the value *tree*.

4.2 TagsMISSkvk: Missing Key If a (key,value) Appears in an Element

The next MR has a similar structure to the previous one but its goal is completely different: we are interested in checking that elements having a certain tag must necessarily have another tag labelled by a certain key. For example, an element with the *(highway, crossing)* tag must include the *crossing* key in another tag according to OSM rules. In addition, the MR also serves to check recommended keys. This is the case, for instance, of the *(amenity, restaurant)* tag for which is recommended to attach the *cuisine* key, or the case of *(highway, residential)* for which is recommended to attach *oneway*. In order to formalise this idea, we will check that given a tag and a key, if there are many elements of the map having this tag but not having the key, then tag and key are not paired very often.

Given two thresholds $\delta, \epsilon \in \mathbb{N}$ and two strings α and β, associated with a key and a value, respectively, let $i_1 = (*, (\alpha, \beta))$ and $O_1 = \mathcal{M}(i_1)$. Given $\alpha' \in keys(O_1) \setminus \{\alpha\}$, let $i_2 = (*, (\alpha,' *))$ and $O_2 = \mathcal{M}(i_2)$.

$$\texttt{TagsMISSkvk}(i_1, i_2, O_1, O_2, \delta, \epsilon) \\ \Updownarrow_{\textbf{def}} \\ |O_1 \cap \overline{O_2}| \geq \delta \Longrightarrow |O_1 \cap O_2| \leq \epsilon$$

Example 3. Consider again the OSM map given in Fig. 2. Let $\alpha = railway$ and $\beta = stop$. Therefore, $i_1 = (*, (railway, stop))$. Again, α' must be set to a key that appears in a tag of the elements having $(railway, stop)$ such that $\alpha' \neq railway$. For example, let $\alpha' = train$ and, therefore, $i_2 = (*, (train, *))$. If we apply the corresponding inputs to the map, we obtain O_1 and O_2. Next, we have to check $\texttt{TagsMISSkvk}(i_1, i_2, O_1, O_2, \delta, \epsilon)$. We have to check whether there are more than δ elements in the map having the $(railway, stop)$ tag but not having the $train$ key. In this case, this holds if $\delta \leq 2$. Now we have to check whether there are at least ϵ elements having the previous tag and key. Since we have three elements in the map fulfilling this property if we set $\epsilon > 2$ then we will not find any error. If we set ϵ to a smaller value, for instance $\epsilon = 2$, then we find two potential errors: those without the $train$ key.

Concerning thresholds, the purpose of δ in this MR is to fix the minimum number of elements having an (α, β) tag such that α' does not appear as key in any tag of the element needed to invoke the MR. Again, if δ is relatively big, then the MR is only applied for relevant features of the map. We have selected $\delta = 20$ for all the experiments in which a large number of elements exist and 10 and 5 for those with a smaller number of elements. In addition, ϵ serves here to fix the number of elements having (α, β) and α'. Thus, the values of ϵ should be greater than the ones used in the previous MR. We have selected 10 for the experiments in which a large number of elements exist and 5 and 2 for those with a smaller number of elements. In both cases, reducing delta and epsilon led to a higher amount of false positives.

The generation of source and follow-up inputs also follows a similar pattern as the one presented in the previous section and, therefore, we omit it.

4.3 TagsCOMPkv: Compatibility Between Keys and Their Associated Values

The goal of the third MR is to detect elements of the map under study that include a (k, v) tag such that k and v should not be paired together. In a *good* map, values should be univocally associated with keys. However, our previous work with OSM [39], [40], [41] showed that this is not always the case. For example, it is usual to find maps with two different elements including, respectively, the tags $(building, construction)$ and $(railway, construction)$. Still, for each value, there should be a *main* key associated with it and, therefore, the number of occurrences of this value associated with other keys should be *small*. In addition, we should exclude values that will certainly be associated with different keys. Specifically, yes, no and numerical values are excluded.

Given two thresholds $\delta, \epsilon \in \mathbb{N}$ and a string α, associated with a key, let $i_1 = (*, (\alpha, *))$ and $O_1 = \mathcal{M}(i_1)$. Given $\beta \in vals(O_1)$, let $i_2 = (*, (*, \beta))$ and $O_2 = \mathcal{M}(i_2)$.

$$\texttt{TagsCOMPkv}(i_1, i_2, O_1, O_2, \delta, \epsilon) \\ \Updownarrow_{\textbf{def}} \\ \left(\begin{array}{c} \beta \notin \mathbb{R} \cup \{yes, no\} \\ \wedge \\ |\{e \in O_1 | (\alpha, \beta) \in tags(e)\}| \geq \delta \end{array} \right) \\ \Downarrow \\ \forall \alpha' \neq \alpha : |\{e \in O_2 | (\alpha,' \beta) \in tags(e)\}| \leq \epsilon$$

If a value is mostly associated to a specific key, then the number of occurrences of this value associated with any other key should be small. Otherwise, the MR will uncover potential errors due to the assignment of the same value to different keys.

The use of thresholds increases the flexibility of this MR. Next, we provide some comments concerning how thresholds should be chosen. First, δ should be taken large enough to discard few occurrences of the (α, β) tag because the MR only applies to predominant (α, β). Second, ϵ should be small enough to discover scarce occurrences. In particular, it should be smaller than δ. In order to facilitate the selection of appropriate values, we have implemented an iterative process that, according to our experiments, provides a good balance between reported actual errors and false positives. The ϵ threshold takes values from zero onwards while δ is always set to $\epsilon + 1$. Note that the number of detected errors cannot increase when we increment ϵ. In fact, the detected errors for smaller values of ϵ –elements which might potentially be false positives– might not be detected for larger values of ϵ, which according to the MR, is due to the same value being used with another key more than ϵ times for a large ϵ. Given a tuple (i_1, i_2, O_1, O_2), we can define its *exit threshold* as the maximum $m \in \mathbb{N}$ for which $\texttt{TagsCOMPkv}(i_1, i_2, O_1, O_2, m+1, m)$ does not hold. It is very likely that potential errors detected by smaller ϵ values but not detected by larger ϵ values (in particular, when ϵ is equal to the exit threshold) are actual errors. In theory, the exit threshold could be undefined (if the MR holds for $\epsilon = 0$). In our experiments, however, this MR always reported potential errors for the (1,0) pair of thresholds.

Example 4. Consider the OSM map given in Fig. 2. Let $\alpha = railway$ and $\beta = stop$. Therefore, $i_1 = (*, (railway, *))$ and $i_2 = (*, (*, stop))$. If we apply the corresponding inputs to the map, we obtain O_1 and O_2. Next, we have to check $\texttt{TagsCOMPkv}(i_1, i_2, O_1, O_2, \delta, \epsilon)$. Depending on the values of the thresholds, we will not detect any potential error (if we choose big values for δ or ϵ). For example, if we set $\delta = 1$ and $\epsilon = 0$ then we will find one potential error: the node with $wrong2$ key. Now, if we set $\delta = 2$ and $\epsilon = 1$ then we will not detect any potential error. Thus, in this example the exit condition of the potential error is 1.

The last step is to generate inputs to *feed* this MR. First, we identify a source input α_0 and let $i_0 = (*, (\alpha_0, *))$. Again, it will be useful to choose a key that appears in a big proportion of tags, so that we (recursively) cover a big portion of the map. Next, we automatically generate λ follow-up inputs, corresponding to values that will conform the second input of our metamorphic relation, as follows. Given $O_0 = \mathcal{M}(i_0)$, we extract all the values $\beta_1, \ldots, \beta_\lambda$ such that each of the pairs $(\alpha_0, \beta_1), \ldots, (\alpha_0, \beta_\lambda)$ is tagging at least δ elements of O_0. In order to speed up the process, we discard numerical and yes/no values.

For each $1 \leq r \leq \lambda$, let $i_0^r = (*, (*, \beta_r))$ and we compute $O_0^r = \mathcal{M}(i_0^r)$. Then, we sequentially check the following λ metamorphic relations:

$$\texttt{TagsCOMPkv}(i_0, i_0^r, O_0, O_0^r, \delta, \epsilon)$$

where $1 \leq r \leq \lambda$. If any of these λ metamorphic relations does not hold, then we have found a failure and could stop. Otherwise, we can iterate the previous process, as follows, to automatically construct λ *families* of source and follow-up inputs. For each $1 \leq r \leq \lambda$, we extract all the keys $\alpha_r^1, \ldots, \alpha_r^{\lambda_r}$, such that for each $1 \leq s \leq \lambda_r$, there are at least δ elements in O_0^r having (α_r^s, β_r) as one of its tags. Let $i_{rs} = (*, (\alpha_r^s, *))$ and compute $O_{rs} = \mathcal{M}(i_{rs})$. Therefore, for each $1 \leq r \leq \lambda$ we can check the following λ_r metamorphic relations:

$$\texttt{TagsCOMPkv}(i_{rs}, i_0^r, O_{rs}, O_0^r, \delta, \epsilon)$$

where $1 \leq s \leq \lambda_r$. If we check these $\sum_{i=1}^{r} \lambda_i$ metamorphic relations and all of them hold, then we can iterate the process by using the newly computed keys and discarding the combinations of pairs (α, β) that have been already exercised.

4.4 TagsName: Compatibility Between Values and Names

Our last MR deals with elements having a *name* that (almost) univocally indicates that this element belongs to a specific category. Consider that an element has the tag $(name, Bailey's\ Hotel)$. Then, we should expect that this element also has a value $hotel$. Again, we will use a threshold because maps may include correct information not following this pattern. For example, if an element has a tag such as $(name, Manchester\ Hoteliers\ Association)$, then it should not include the before mentioned $hotel$ value. In addition, we should apply our MR only if the value associated with the name is a sensible string to be associated with a *generic* value. Therefore, we will check that there are *many* elements that use it as a value.

Given two thresholds $\delta, \epsilon \in \mathbb{N}$ and a string β, associated with a value, let $i_1 = (*, (*, \beta))$, $i_2 = (*, (name, *\beta*))$, $O_1 = \mathcal{M}(i_1)$ and $O_2 = \mathcal{M}(i_2)$.

$$\texttt{TagsName}(i_1, i_2, O_1, O_2, \delta, \epsilon)$$
$$\Updownarrow_{\textbf{def}}$$
$$|O_1| \geq \delta \Rightarrow |O_2 \cap \overline{O_1}| \leq \epsilon$$

Example 5. Consider the OSM map given in Fig. 2. Let $\beta = station$. Therefore, $i_1 = (*, (*, station))$ and $i_2 = (*, (name, *station*))$. If we apply the corresponding inputs to the map, we obtain O_1 and O_2. Next, we have to check $\texttt{TagsName}(i_1, i_2, O_1, O_2, \delta, \epsilon)$. As before, if we choose big values for δ or ϵ we will not detect any potential error. For example, if we set $\delta = 1$ and $\epsilon = 1$, then we have one potential error.

In practice, similarly to the case of the first two MRs, δ should be selected according to the frequency of use of β and values for ϵ should be taken accordingly to δ values.

In contrast to the previously presented MRs, we cannot automatically iterate the process of producing inputs because the tester plays a fundamental role in selecting *sensible* values. Note that this situation is quite usual in MT. Therefore, the tester will choose a string β_0 to be used as source input, consider $i_0 = (*, (*, \beta_0))$ and $i_0' = (*, (name, *\beta_0*))$, compute $O_0 = \mathcal{M}(i_0)$ and $O_0' = \mathcal{M}(i_0')$, and evaluate whether

$$\texttt{TagsName}(i_0, i_0,' O_0, O_0,' \delta, \epsilon)$$

holds. If an error is not detected, then the tester will choose another string to be used as source input and iterate the process.

4.5 Implementation Details

We have implemented our MRs and applied them on different OSM maps. Next, we briefly describe the main features of the programming language that we have used and present sample codes. Our MRs have been implemented in the XML Database Query Language XQuery [44], [45]: a functional language based on FLWOR ("For, Let, Where, Order by, Return") expressions and using XPath to select nodes from the analysed XML trees. The main reasons to choose this language is our expertise (we have extensively used it in previous work), OSM maps can be exported into XML and, in addition, working with a declarative programming language strongly simplifies the implementation of our MRs.

Thanks to its higher order capabilities, XQuery allows inputs application to be defined as a higher order function. For instance, if we have an input $i = (T, C)$, then the application of this input to a map, $\mathcal{M}(i)$, is given in XQuery by the code:

```
declare function mt:M($T,$C)
{
filter($map/osm/*,
 function($e){mt:type($e,$T) and $C($e)})
};}
```

The inputs of the different MRs are transformed into XQuery Boolean conditions by using XQuery "some" statements and XPath expressions. For example, the *translation* of the input condition $C = \{(name, \beta), (highway, *)\}$ is given by the code:

```
some $z in $x/tag satisfies $z[@k=''name''and
 @v=$beta] and (some $z in $x/tag satisfies $z
[@k=''highway''])
```

where x/tag stands for the tags of x and $(@k, @v)$ corresponds to the (key, value) pairs of a tag.

MRs are mapped into XQuery Boolean conditions using XQuery "some" and "every" statements, as well as union/intersection operators. For example, the `TagsCOMPkvk` MR can be defined as:

```
if (count($O1) >= $delta)
then
count($O1 intersect $O2)>=$epsilon
else true()
```

Finally, we use XQuery to store and access the XML representation of OSM maps. Since XML documents are automatically indexed by XQuery, we will have an efficient query and access to the items of OSM maps.

All the code used to perform the experiments, in particular, the implementation of our MRs, can be found at https://github.com/jalmenUAL/MT-OSM.

5 Experiments and Empirical Evaluation

In this section we report on the results of the experiments that we have performed and present representative errors that have been found and, very likely, might appear in other OSM maps. First, we would like to pose different research questions.

5.1 Research Questions

The final goal of our approach is to find errors in OSM. Therefore, despite using a different approach, based on thresholds, and targeting different types of errors, geometries vs. labels, it is only natural to repeat the same research questions as in our previous work [9]. First, we would like to evaluate the scalability of our framework.

Research Question 1. *Does our framework scale well?*

Second, we would like to assess how good is our framework in detecting errors, in this case, in OSM.

Research Question 2. *Is our framework an effective way to find errors in maps?*

Third, we would like to evaluate how good is our framework in comparison with other tools and systems.

Research Question 3. *How does our framework compare to other approaches?*

However, we have a new characteristic in our framework, the use of thresholds, that poses a new question.

Research Question 4. *How sensitive are MRs to the choice of thresholds?*

5.2 Experiments

We have conducted experiments in four areas of different cities: Madrid (Barrio de Salamanca), London (Kensington), Sao Paolo (Republica) and Nairobi (Park Road). The initial source inputs, that is, the *seeds* of the process, have been chosen to cover common tags. Specifically, the ones corresponding to the *building* (6%) and *highway* (2,6%) keys, but also other tags less frequently used such as *amenity* (0,24%) and *tourism* (0,03%). The total number of analysed tags in each city is as follows:

$\delta,\ \epsilon$	OSM	Key	Value	Conflict	Osmose	Reported
10,2	Link	building	yes	**country**	Link	No
10,2	Link	building	yes	**diplomatic**	Link	No
10,2	Link	building	yes	**target**	Link	No
10,2	Link	building	yes	**office**	Link	No
10,2	Link	building	yes	website	-	-
10,2	Link	building	yes	roof:colour	-	-
10,2	Link	building	yes	roof:orientation	-	-
10,2	Link	building	yes	roof:shape	-	-
10,2	Link	building	yes	**social_facility**	Link	No
10,2	Link	building	yes	**social_facility:for**	Link	No
10,2	Link	building	yes	addr:state	-	-
10,2	Link	building	yes	**brand**	Link	No
10,2	Link	building	yes	**brand:ca**	Link	No
10,2	Link	building	yes	**brand:es**	Link	No
10,2	Link	building	yes	**brand:wikidata**	Link	No
10,2	Link	building	yes	**brand:wikipedia**	Link	No
10,2	Link	building	yes	name:ca	-	-
10,2	Link	building	yes	official_name	-	-
10,2	Link	building	yes	source	-	-
10,2	Link	building	yes	addr:unit	-	-
10,2	Link	building	yes	**phone**	Link	No
10,2	Link	building	yes	fixme	-	-
10,2	Link	building	yes	**fee**	Link	No
10,2	Link	building	yes	**parking**	Link	No
10,2	Link	building	yes	**access**	Link	No
10,2	Link	building	yes	**surface**	Link	No
10,2	Link	building	yes	**access**	Link	No
10,2	Link	building	yes	**level**	Link	No
10,2	Link	building	yes	**lit**	Link	Yes2
10,2	Link	building	yes	**operator**	Link	No
10,2	Link	building	yes	**sport**	Link	No
10,2	Link	highway	traffic_signals	**bus**	Link	No
10,2	Link	highway	traffic_signals	**public_transport**	Link	No
10,2	Link	highway	crossing	**bicycle**	Link	No
10,2	Link	highway	crossing	traffic_signals:sound	-	-
10,2	Link	highway	crossing	traffic_signals:vibration	-	-
10,2	Link	highway	secondary	sidewalk	-	-
10,2	Link	highway	secondary	**psv:lanes**	Link	No
10,2	Link	highway	residential	cycleway:right	-	-
10,2	Link	highway	residential	**source:maxspeed**	Link	No
10,2	Link	highway	residential	lit	-	-
10,2	Link	highway	service	source:date	-	-
10,2	Link	highway	service	**layer**	Link	No
10,2	Link	highway	service	**level**	Link	No
10,2	Link	amenity	pharmacy	**phone**	Link	No
10,2	Link	amenity	pharmacy	**email**	Link	No
10,2	Link	amenity	pharmacy	wheelchair	-	-
10,2	Link	amenity	pharmacy	**addr:ward**	Link	Yes3
10,2	Link	amenity	restaurant	toilets:wheelchair	-	-
10,2	Link	amenity	restaurant	delivery	-	-
10,2	Link	amenity	restaurant	outdoor_seating	-	-
10,2	Link	amenity	restaurant	description	-	-
10,2	Link	amenity	bank	old_name	-	-
10,2	Link	amenity	bank	**name:1991-2010**	Link	No
10,2	Link	amenity	bank	short_name	-	-
10,2	Link	amenity	bank	opening_hours	-	-
10,2	Link	amenity	bank	**building**	Link	Yes3
4,2	Link	tourism	hotel	internet_access	-	-
4,2	Link	tourism	hotel	addr:country	-	-

Fig. 3. Application of `TagsCOMPkvk` (**Madrid**). **Actual errors in bold**. Total: 59. False Positives: 26. Osmose: 3.

	building	*highway*	*amenity*	*tourism*
Madrid	658	536	200	10
London	476	382	119	16
Sao Paulo	608	247	113	20
Nairobi	882	339	46	6

Figs. 3, 4, 5, 6, 7, 8 and 9 summarise the most relevant categories in which our MRs were able to find errors in the analysed maps. We indicate whether the detected errors are identified in *Osmose*. In this case, we have the following abbreviations for different *tag conflicts:*

Yes1	Tourism with missing or wrong value
Yes2	Building with missing or wrong tag
Yes3	Amenity with missing or wrong tag
Yes4	Highway with missing or wrong tag

δ, ϵ	OSM	Key	Value	Conflict	Osmose	Reported
10,2	Link	building	yes	**phone**	Link	No
10,2	Link	building	yes	photo	-	-
10,2	Link	building	yes	start_date	-	-
10,2	Link	building	yes	**landuse**	Link	No
10,2	Link	building	yes	building:flats	-	-
10,2	Link	building	yes	**tourism**	Link	No
10,2	Link	building	yes	source:name	-	-
10,2	Link	building	yes	**phone**	Link	No
10,2	Link	building	yes	wikidata	-	-
10,2	Link	building	yes	wikipedia	-	-
10,2	Link	building	yes	**toilets:access**	Link	No
10,2	Link	building	yes	contact:pinterest	-	-
10,2	Link	building	yes	**email**	Link	No
10,2	Link	building	yes	**internet_access**	Link	No
10,2	Link	building	yes	**operator**	Link	No
10,2	Link	building	yes	**smoking**	Link	No
10,2	Link	building	yes	**rooms**	Link	No
10,2	Link	building	yes	**tourism**	Link	No
10,2	Link	building	yes	**cuisine**	Link	No
10,2	Link	building	yes	**disused:amenity**	Link	No
10,2	Link	building	yes	**layer**	Link	No
10,2	Link	building	yes	**architect**	Link	No
10,2	Link	building	yes	building:wikipedia	-	-
10,2	Link	building	yes	listed_status	-	-
10,2	Link	building	yes	**wheelchair**	Link	No
10,2	Link	building	yes	**tourism**	Link	No
10,2	Link	building	yes	**shelter_type**	Link	No
10,2	Link	building	yes	**power**	Link	No
10,2	Link	building	yes	**brand**	Link	No
10,2	Link	building	yes	**tourism**	Link	No
10,2	Link	building	terrace	addr:postcode	-	-
10,2	Link	building	terrace	**brand**	Link	No
10,2	Link	building	terrace	**email**	Link	No
10,2	Link	building	terrace	**internet_access:fee**	Link	No
10,2	Link	building	terrace	name	-	-
10,2	Link	building	terrace	**opening_hours**	Link	No
10,2	Link	building	terrace	**payment:cash**	Link	No
10,2	Link	building	terrace	**payment:mastercard**	Link	No
10,2	Link	building	terrace	**payment:visa**	Link	No
10,2	Link	building	terrace	**phone**	Link	No
10,2	Link	building	terrace	**rooms**	Link	No
10,2	Link	building	terrace	**tourism**	Link	No
10,2	Link	highway	bus_stop	note	-	-
10,2	Link	highway	bus_stop	**bus**	Link	No
10,2	Link	highway	crossing	tactile_paving	-	-
10,2	Link	highway	service	bicycle	-	-
10,2	Link	highway	service	**unsigned**	Link	No
10,2	Link	highway	service	note	-	-
10,2	Link	highway	service	covered	-	-
10,2	Link	highway	service	fixme	-	-
10,2	Link	highway	service	lit	-	-
10,2	Link	highway	residential	ref	-	-
10,2	Link	highway	residential	**rcn**	Link	No
10,2	Link	highway	residential	layer	-	-
10,2	Link	highway	residential	covered	-	-
10,2	Link	highway	residential	tunnel	-	-
10,2	Link	highway	residential	cycleway	-	-
10,2	Link	highway	trunk	**operator**	Link	No
10,2	Link	highway	footway	**indoor**	Link	No
10,2	Link	highway	footway	**created_by**	Link	No
10,2	Link	highway	footway	**peacock**	Link	No
10,2	Link	highway	footway	crossing:island	-	-
10,2	Link	highway	footway	tactile_paving	-	-
10,2	Link	amenity	parking	**building**	Link	No
10,2	Link	amenity	parking	park_ride	-	-
4,2	Link	tourism	hotel	addr:city	-	-
4,2	Link	tourism	hotel	**building**	Link	No
4,2	Link	tourism	hotel	**building:colour**	Link	No
4,2	Link	tourism	hotel	**building:levels**	Link	No
4,2	Link	tourism	hotel	contact:instagram	-	-
4,2	Link	tourism	hotel	source	-	-
4,2	Link	tourism	hotel	internet_access:fee	-	-
4,2	Link	tourism	hotel	**phone**	Link	No
4,2	Link	tourism	hotel	**building**	Link	No
4,2	Link	tourism	hotel	**building:levels**	Link	No

Fig. 4. Application of `TagsCOMPkvk` (**London**). **Actual errors in bold**. Total: 73. False Positives: 30. Osmose: 0.

Our approach shows very satisfactory results because we are able to identify a big number of errors in the analysed maps. After a subsequent analysis, taking into account the recommendations given by the OSM community, we detected several false positives. They were mainly identified by our MRs due to few occurrences of the selected keys in the elements of the considered areas.

5.2.1 *Application of* `TagsCOMPkvk`

Figs. 3, 4, 5 and 6 show the results obtained from the application of the `TagsCOMPkvk` MR. The δ, ϵ column shows the considered thresholds, the OSM column links to OSM data of the identified element, the Key and Value columns show (α, β), the Conflict column includes the value of α' causing the MR failure, the Osmose column links to the *Osmose* web site in which the element is located and the Reported column shows whether or not the error is reported by *Osmose*.

δ, ϵ	OSM	Key	Value	Conflict	Osmose	Reported
10,2	Link	building	yes	building:levels	-	-
10,2	Link	building	yes	official_name	-	-
10,2	Link	building	yes	**operator**	Link	No
10,2	Link	building	yes	**tourism**	Link	No
10,2	Link	building	yes	**office**	Link	Yes2
10,2	Link	building	yes	**layer**	Link	No
10,2	Link	building	yes	**atm**	Link	No
10,2	Link	building	yes	**brand**	Link	No
10,2	Link	building	yes	**shop**	Link	No
10,2	Link	highway	traffic_signals	**uncontrolled**	Link	Yes4
10,2	Link	highway	traffic_signals	crossing_ref	-	-
10,2	Link	highway	pedestrian	short_name	-	-
10,2	Link	highway	pedestrian	area	-	-
10,2	Link	highway	pedestrian	**internet_access**	Link	No
10,2	Link	highway	pedestrian	**internet_access:fee**	Link	No
10,2	Link	highway	pedestrian	**internet_access:operator**	Link	No
10,2	Link	highway	pedestrian	post_code	-	-
10,2	Link	highway	pedestrian	**internet_access:operator**	Link	No
10,2	Link	highway	pedestrian	post_code	-	-
10,2	Link	highway	pedestrian	tunnel	-	-
10,2	Link	highway	residential	lanes	-	-
10,2	Link	highway	residential	parking:line:left	-	-
10,2	Link	highway	secondary	lanes:forward	-	-
10,2	Link	highway	secondary	cycleway	-	-
10,2	Link	highway	secondary	busway:left	-	-
10,2	Link	highway	secondary	lanes:bus	-	-
10,2	Link	highway	secondary	**lanes:bus:conditional**	Link	Yes4
10,2	Link	highway	footway	tunnel	-	-
10,2	Link	highway	footway	noname	-	-
10,2	Link	highway	footway	width	-	-
10,2	Link	amenity	restaurant	alt_name	-	-
10,2	Link	amenity	restaurant	source	-	-
10,2	Link	amenity	restaurant	**phone**	Link	No
10,2	Link	amenity	restaurant	name:ko	-	-
10,2	Link	amenity	restaurant	takeaway	-	-
10,2	Link	amenity	restaurant	name:ja	-	-
10,2	Link	amenity	restaurant	**location**	Link	No
10,2	Link	amenity	restaurant	**phone**	Link	No
10,2	Link	amenity	restaurant	wikipedia	-	-
10,2	Link	amenity	restaurant	**tourism**	Link	No
10,2	Link	amenity	restaurant	start_date	-	-
10,2	Link	amenity	restaurant	**building:levels**	Link	Yes3
10,2	Link	amenity	restaurant	**building:part**	Link	Yes3
10,2	Link	amenity	restaurant	**min_height**	Link	No
10,2	Link	amenity	restaurant	**height**	Link	No
10,2	Link	amenity	fast_food	fixme	-	-
10,2	Link	amenity	bank	name:pt	-	-
10,2	Link	amenity	bank	**building**	Link	No
6,2	Link	tourism	hotel	description	-	-
6,2	Link	tourism	hotel	**phone**	Link	No
6,2	Link	tourism	hotel	**email**	Link	No
6,2	Link	tourism	hotel	name:ja	-	-
6,2	Link	tourism	hotel	website	-	-

Fig. 5. Application of `TagsCOMPkvk` (**Sao Paolo**). **Actual errors in bold**. Total: 53. False Positives: 29. Osmose: 5.

Values in bold are actual errors. In summary, from 198 errors found, 88 are false positives. Thus, 56% of precision has been reached in our experiments. Additionally, *Osmose* has reported only 11 of the real errors.

In the case of Madrid, the application of the MR using *building* as seed detected 31 potential errors, being 21 of them real errors. For instance, the simultaneous use of the *(building, yes)* tag and a key such as *diplomatic*, *social_facility* or *surface*. False positives were found, for instance, because of the use of multiple, but scarce, representations of the *name* feature, for instance, *name:ca* and *official_name*. Regarding the use of the *highway* key, 7 out of 13 possible errors correspond to actual errors. An example of real errors is the combination of the *(highway, service)* tag and the *level* key because this key should be used for *ways* representing areas like buildings. Another detected error corresponds to the use of the *(highway, secondary)* tag and the *psv:lanes* key in the same element. The latter is deprecated. Wrong keys, such as *source:*

δ, ϵ	OSM	Key	Value	Conflict	Osmose	Reported
10,2	Link	building	yes	**healthcare**	Link	No
10,2	Link	building	yes	**office**	Link	Yes2
10,2	Link	building	yes	**amenity**	Link	No
10,2	Link	building	yes	**operator**	Link	No
10,2	Link	building	yes	**postcode**	Link	Yes2
10,2	Link	highway	residential	**created_by**	Link	No
10,2	Link	highway	primary	**layer**	Link	No
10,2	Link	highway	primary	junction	-	-
4,2	Link	amenity	place_of_worship	denomination	-	-
4,2	Link	amenity	place_of_worship	**addr:housenumber**	Link	Yes1
4,2	Link	amenity	school	**building**	Link	No
4,2	Link	amenity	parking	ref	-	-
4,2	Link	amenity	parking	**operator**	Link	No

Fig. 6. Application of `TagsCOMPkvk` (**Nairobi**). **Actual errors in bold**. Total: 13. False Positives: 3. Osmose: 3.

maxspeed, were also detected. False positives are due to scarcely used keys such as *sidewalk* or *lit*. In the case of the *amenity* key, 8 out of 13 possible errors are false positives. False positives were due again, for instance, to scarcely used keys such as *description* together with the *(amenity, restaurant)* tag. In addition, unknown keys, such as *addr:ward* and *name:1991-2010* were appropriately reported. With regard to the *tourism* key, 2 false positives were uncovered.

With regard to London and the *building* key, 42 potential errors were detected, being 31 of them actual errors. For instance, the simultaneous use of the *(building, yes)* tag and the *shelter_type* key or the *(building, yes)* tag and the *power* key. False positives were due to a low level of occurrences of the corresponding combinations. As an example, the *listed_status* key and the *(building, yes)* tag, because *listed_status* is only used in the U.K. In the case of the *highway* key, 7 errors and 14 false positives were reported. For example, the combination of *(highway, bus_stop)* and *bus* is incorrect. The correct combination would be *(public_transport, bus_stop)* and *bus*. Another example is related to the coexistence of the *(highway, footway)* tag and the *indoor* key in an element, because *indoor* must be used only for interior corridors and rooms. Incorrect keys, like *peacock*, were also found. Regarding false positives, once again, they were due to combinations scarcely used, like the occurrence of the *(highway, service)* tag and the *covered* key in the same element. In the case of the *amenity* key, 2 potential errors were found, but only 1 of them corresponded to a real error: the use of the *(amenity, parking)* tag and the *building* key in the same element. The application of the *tourism* key exposed 6 actual errors and 4 false positives. The errors were related to combinations of this key and building properties such as *building:levels* and *building:colour*. The most remarkable false positive corresponds to the *source* and *addr:city* keys.

Regarding Sao Paolo and the use of the *building* key as seed, this MR detected 7 real errors and 2 false positive were reported. One of the detected errors corresponds to the combination of the *(building, yes)* tag and the *historic* or *layer* keys. The application of the MR using as source input the *highway* key uncovered 6 errors, while marking 15 false positives. An unknown tag, such as *uncontrolled*, or wrong combinations, such as *(highway, pedestrian)* and the *internet_access* key, are examples of real errors. False positives are related to the use of multiple and scarce keys for names such as *short_name* or non usual keys such as *tunnel* or *post_code*. The use of the *amenity* key produced 15 potential errors, being 9 of them actual errors. In this group we can find combinations of *(amenity, restaurant)* with the *min_height* or *height* keys, which are typical properties of buildings. Once again, different keys for names caused false positives. Finally, the *tourism* key reported 5 errors and 3 false positive.

In the case of Nairobi and the *building* key, 5 errors were exposed. The use of unknown keys, such as *postcode*, and wrong combinations such as *(building, yes)* and the *operator* key are the most relevant. The use of the *highway* key reported 2 actual errors The incorrect use of the *layer* key together with the *(highway, primary)* tag is remarkable, since *layer* can be combined only with the *tunel*, *bridge* and *covered* keys or the *(highway, steps)* and *(highway, elevator)* tags. The *amenity* key produced 3 real errors and 2 false positives. It is worth pointing out the wrong combination of *(amenity, place_of_worship)* and the *addr:housenumber* key.

δ, ϵ	Total Miss.	Total	Key	Value	Missing	Reported
			Madrid			
20,10	206	230	building	yes	addr:city	-
20,10	212	230	building	yes	addr:housenumber	-
20,10	212	230	building	yes	addr:street	-
20,10	366	420	building	apartments	building:levels	-
20,10	391	420	building	apartments	addr:city	-
20,10	60	90	highway	traffic_signals	**traffic_signals**	49
20,10	29	145	highway	crossing	**crossing**	28
20,10	22	37	highway	tertiary	surface	-
20,10	34	140	highway	footway	**footway**	12
10,5	23	46	amenity	restaurant	cuisine	-
10,5	13	46	amenity	restaurant	addr:housenumber	-
10,5	12	46	amenity	restaurant	addr:street	-
10,5	20	46	amenity	restaurant	addr:city	-
10,5	35	46	amenity	restaurant	contact:website	-
10,5	21	46	amenity	restaurant	addr:postcode	-
10,5	30	46	amenity	restaurant	addr:country	-
10,5	34	46	amenity	restaurant	contact:phone	-
10,5	12	19	amenity	pharmacy	addr:city	-
			London			
20,10	32	65	building	yes	addr:city	-
20,10	34	65	building	yes	building:levels	-
20,10	39	65	building	yes	addr:housenumber	-
20,10	26	65	building	yes	source	-
20,10	37	65	building	yes	addr:street	-
20,10	20	54	building	apartments	building:flats	-
20,10	27	54	building	apartments	source	-
20,10	130	177	building	house	source	-
20,10	24	42	building	semidetached_house	source	-
20,10	57	78	highway	residential	lit	-
20,10	50	78	highway	residential	surface	-
20,10	47	78	highway	residential	oneway	-
20,10	60	104	highway	footway	**footway**	36
20,10	89	104	highway	footway	foot	-
20,10	32	55	highway	service	**service**	0
20,10	33	55	highway	service	source	-
10,5	17	33	amenity	bench	**backrest**	0
			Sao Paolo			
20,10	551	617	building	yes	building:part	-
20,10	34	71	highway	crossing	**crossing**	25
20,10	12	32	highway	traffic_signals	**traffic_signals**	10
20,10	28	44	highway	footway	**footway**	9
10,5	12	21	amenity	bank	brand	-
10,5	12	21	amenity	bank	brand:wikidata	-
10,5	14	20	amenity	restaurant	**phone**	0
5,2	6	10	tourism	hotel	internet_access	-
5,2	5	10	tourism	hotel	phone	-
5,2	7	10	tourism	hotel	source	-
5,2	5	10	tourism	hotel	entrance	-
			Nairobi			
20,10	832	873	building	yes	name	-
20,10	85	120	highway	residential	name	-
20,10	108	120	highway	residential	source	-
20,10	76	120	highway	residential	oneway	-
20,10	21	33	highway	unclassified	surface	-
20,10	43	117	highway	service	**service**	0

Fig. 7. Application of `TagsMISSkvk`. **Mandatory in bold**. Total Missing Mandatory: 363. Osmose: 169.

5.2.2 Application of `TagsMISSkvk`

Fig. 7 shows the results of the application of the `TagsMISSkvk` MR to the considered maps. Again, the δ, ϵ column shows the considered thresholds, the Total Miss column includes the number of elements without the considered α', the Total column shows the total number of elements with (α, β), the Key and Value columns show (α, β) and Missing shows α'. As usual, the Reported column shows whether or not the error is reported by *Osmose*. In this case, values in bold are mandatory keys while non bold values are recommended keys. Here the Osmose link is not shown because we report elements in an aggregated way. In summary, from 4337 errors found, 363 are actual errors – they are mandatory– and 3974 are recommended combinations. None of them are false positives. Additionally, *Osmose* has reported only 169 of the mandatory ones.

All the errors reported by the application of this MR using the *building* key in the map of Madrid correspond to recommended combinations. The lack of the *addr:street* and *addr:city* keys with either *(building, yes)* or *(amenity, restaurant)* is a typical example. In the case of *highway*, 3 missing mandatory combinations were detected: *(highway, traffic_signals)* with the *traffic_signals* key, *(highway, crossing)* with

OSM	Key	Value	Conflict	Exit	Osmose	Reported
Madrid						
Link	building	**college**	amenity	1	Link	No
Link	building	**hospital**	amenity	2	Link	No
Link	building	**hospital**	amenity	2	Link	No
Link	amenity	**dentist**	healthcare	1	Link	No
Link	amenity	**embassy**	diplomatic	1	Link	Yes3
London						
Link	building	**kindergarten**	amenity	1	Link	No
Link	building	**school**	amenity	1	Link	No
Link	building	construction	landuse	1	-	-
Link	amenity	**dentist**	healthcare	1	Link	No
Link	amenity	**pharmacy**	healthcare	1	Link	No
Link	tourism	**Natural England**	CREATEDBY	1	Link	No
Sao Paolo						
Link	amenity	**bicycle_rental**	disused:amenity	1	Link	No
Nairobi						
Link	building	**hotel**	tourism	2	Link	Yes1
Link	building	**hotel**	tourism	2	Link	No
Link	building	**school**	amenity	2	Link	No
Link	building	**school**	amenity	2	Link	No
Link	amenity	**dentist**	healthcare	1	Link	No
Link	amenity	**swimming_pool**	leisure	2	Link	Yes3
Link	amenity	**swimming_pool**	leisure	2	Link	Yes3
Link	tourism	hotel	building	2	-	-
Link	tourism	hotel	building	2	-	-

Fig. 8. Application of `TagsCOMPkv`. **Actual errors in bold**. Total: 21. False Positives: 3. Osmose: 4.

the *crossing* key and *(highway, footway)* with the *footway* key. All the results reported by using the *amenity* key were recommended combinations. For instance, as said before, it would be desirable that elements labelled with the *(amenity, restaurant)* tag would include the *cuisine* key.

In the case of London and the *building* key, all the reported errors refer to recommended combinations. For example, a high percentage of the elements labelled with this key do not present a tag associated with the *source* key, a combination recommended by the OMS community. The application of the MR was able to detect missing mandatory combinations associated with the use of the *highway* key, such as *footway* and *service*. The application of the *amenity* key as source input also reported the lack of mandatory combinations such as *(amenity, bench)* and the *backrest* key.

In the case of Sao Paolo and the *building* key, a recommended combination is the *building:parts* key together with the *(building, yes)* tag. Again, in the case of the *highway* key, for the *traffic_signals* and *crossing* values, the mandatory keys *traffic_signals* and *crossing*, respectively, were missing in some elements, as well as the *footway* key for elements that present the *(highway, footway)* tag. The *amenity* key reports missing mandatory combinations. Here, there is a remarkable case: the occurrence of the *(amenity, restaurant)* tag but the lack of the *phone* key. Finally, *tourism* only reports recommended combinations.

If we consider the map of Nairobi and the *(building,yes)* tag, we also found some missing recommended combinations such as *name*. The application of the *highway* key to this map allowed us to uncover a missing mandatory combination: *(highway, service)* and the *service* key. All the other results correspond to the lack of recommended combinations for *(highway,residential): oneway*, *name* and *source*; and for *(highway,unclassified): surface*. The *amenity* key did not reveal any error due to the low level of occurrences of this tag in the elements.

5.2.3 Application of `TagsCOMPkv`

Fig. 8 shows the results of the application of the `TagsCOMPkv` MR. The Exit column shows the *exit threshold* while the rest of the columns show the same information as in Fig. 3. In summary, from 21 potential errors found by the application

δ, ϵ	Key	OSM	Name	Missing Tags
Madrid				
1,1	park	Link	Parking bicis Alcántara	**(leisure,park)**
London				
1,1	park	Link	Parks Police	(leisure,park)
1,1	park	Link	Holland Park	**(leisure,park)**
1,1	park	Link	Holland Park Road	(leisure,park)
1,1	park	Link	Park Close	(leisure, park)
1,1	park	Link	Holland Park Nets	(leisure,park)
Sao Paolo				
1,1	park	Link	Republica Park Hotel	(leisure,park)
Nairobi				
1,1	hotel	Link	Oloibon hotel	**(tourism,hotel)**
1,1	park	Link	UHURU PARK	(leisure,park)
1,1	park	Link	Park Road	(leisure, park)
1,1	park	Link	Park Road Hotel	(leisure,park)
1,1	restaurant	Link	Pool Restaurant	**(amenity,restaurant)**
1,1	restaurant	Link	Simmers Bar Restaurant	**(amenity,restaurant)**

Fig. 9. Application of `TagsName`. **Actual errors in bold**. Total: 13. False Positives: 8. Osmose: 0.

of our MR, only 3 are false positives. It is worth to emphasise that *Osmose* has reported only 4 of the real errors.

This MR allows the detection of a typical mistake in OSM: the same values are used for the keys *building* and *amenity/ tourism*. For instance, the value *hospital* is associated with the keys *building* and *amenity*. However, OSM community guidelines strongly recommend the use of *amenity* and *tourism* instead of *building* in these cases. It is worth to point out that the *amenity* key covers many values corresponding to facilities for visitors and residents, such as toilets, telephones, banks, pharmacies, prisons and schools. However, *building* is intended only to describe a building from a physical point of view.

In the case of Madrid we found the following errors: (1) the previously mentioned case related to the keys *building* and *amenity*; (2) deprecated tags such as *(amenity, embassy)*, for which the recommended labelling is *(diplomatic, embassy)*. Finally, let us remark that the applications of this MR using *tourism* and *amenity* revealed exactly the same errors. This is due to the *recursive* nature in which follow-up inputs are produced.

In London we detected: (1) the previously mentioned case of simultaneous use of the *building* and *amenity* keys in tags such as *(building, kindergarten)* and *(amenity, kindergarten)*; (2) the use of a *(healthcare, dentist)* tag even though the key *healthcare* is only recommended for the value *clinic*; (3) a misspelling of *"CREATEDBY"* in the case of the *tourism* key. Finally, a false positive caused by the coexistence of the *(building, construction)* and *(landuse, construction)* tags was reported: it is allowed to use the *construction* value with both keys.

This MR revealed only one error in the case of Sao Paolo: prefix *"disused"* used together with the original label.

Finally, Nairobi also presents co-occurrences of the *building* and *tourism* keys with the same value, in this case *hotel*, being only allowed for *building*. Another example corresponds to the *amenity* and *leisure* keys, both of them associated with the value *swimming_pool*. Both combinations were allowed up to 2019, being normative the use of the *leisure* key from this date.

5.2.4 Application of `TagsName`

Fig. 9 presents the results obtained from the application of the `TagsName` MR. In addition to columns used in previous sections, the Name column includes the *name* of the reported element while Missing tags shows the missing tag. We do not include *Osmose* links because *Osmose* does not report this kind of errors. In this case, 8 out of 19 detected potential

TABLE 1
Scalability Datasets

BoundingBox	Size (MB)	Elements	Nodes	Ways
-3.67064, 40.42702, -3.65900, 40.43178	2.9	8743	7587	1156
-3-67870, 40.42300, -3.65540, 40.43260	8.6	33500	28718	4782
-3.70090, 40.41370, -3.65430, 40.43270	15.9	109712	95431	14281
-3.72420, 40.40020, -3.63110, 40.44220	59.4	427036	369592	57444
-3.77070, 40.38510, -3.58450, 40.46130	123.5	1000720	859882	140838
-3.86380, 40.34710, -3.49120, 40.49940	251.9	2160991	1851862	309129
-4.04980, 40.27060, -3.30480, 40.57540	447.8	3959529	3408487	551042
-4.42200, 40.11800, -2.93200, 40.72750	769.8	6088426	5318702	769724

TABLE 2
Scalability Tests

	`TagsCOMPkvk`		`TagsMISSkvk`		`TagsCOMPkv`		`TagsName`	
Size (MB)	RT (ms)	#	RT (ms)	#	RT (ms)	#	RT (ms)	#
2.9	12,710.61	11	4,446.76	1039	4,017.09	3	83.8	0
8.6	49,992.01	36	38,418.04	5,194	1,558.43	7	189.9	0
15.9	135,675.32	77	107,967.54	14,654	7,734.69	16	394.88	2
59.4	359,665.32	99	298,988.83	39,880	182,487.56	42	1,281.67	4
123.5	1,077,677.14	176	1,034,654.09	82,960	589,677.03	59	2,992.08	6
251.9	2,234,554.83	243	2,688,767.43	132,845	1,488,765.32	85	5,996.24	10
447.8	Stack overflow		Stack overflow		Stack overflow		10,504.61	17
769.8	Stack overflow		Stack overflow		Stack overflow		15,777.79	28

errors were false positives. Most of the reported false positives were associated with elements in which the *(leisure, park)* tag is missing. Usually, there are other features adopting the name "Park" because of its vicinity. However, *hotels* and *restaurants* are actual errors in most of the cases. Finally, the customisation of δ and ϵ is given, in both cases, by 1.

5.3 Answer to the Research Questions

In view of these results, we can provide a positive answer to our Research Question 2: our approach is very effective to find errors in OSM. However, we must mention that three of our MRs, in addition to real errors, collected a non-negligible amount of false positives. We will come back to this issue when we discuss Research Question 4.

Even if our framework were able to reveal errors, its usefulness would be strongly reduced if it could not cope with *standard* maps. Our framework implements the same solution as other tools analysing OSM: big areas are divided into small/medium bounding areas and they are separately analysed. In order to answer our Research Question 1, we performed an additional experiment: we applied our MRs to different areas of Madrid. Table 1 shows the coordinates of the chosen area, the size of the corresponding map and the number of elements included in the map (nodes and ways). Table 2 shows running times and number of errors detected by each MR. Some MRs require more time than others, but they run in reasonable time for sizes smaller than 60 megabytes. Nevertheless, we have to mention that the experiments with the biggest areas did not conclude for three of our MRs. Therefore, we can affirmatively answer to our Research Question 1 but we should add a reservation: our approach scales well but bigger areas should be further split.

Research Question 3 considers whether our framework obtains better results than other alternative tools. In order to perform this comparison, we compared our framework against *Osmose*, the *de facto* standard OSM error checker. As we have mentioned during the analysis of the results of our experiments, *Osmose* has not reported the big majority of the errors found by our MRs. Although *Osmose* does not work with the errors targeted by $\texttt{TagsName}^{\delta,\epsilon}$, it does work with the other three categories of errors. In fact, only 184 errors from 496 actual errors found by our first three MRs are reported by *Osmose* (around 31%). Therefore, we can safely conclude that our framework finds many unreported errors.

Thresholds are a novel component and, as such, their impact on defining useful MRs needed to be studied. Ideally, the specific choice of thresholds should be (relatively) independent: all the thresholds should induce similar MRs, with similar capabilities to detect real errors and similar amounts of false positives. Unfortunately, this is not the case and the answer to Research Question 4, how sensitive are MRs to the choice of thresholds?, is "very sensitive". However, we provide specific values for different source inputs for three of our MRs. In addition, our framework provides an automatic mechanism to iteratively compute the thresholds corresponding to the third MR, `TagsCOMPkv`, which provides a good trade-off between increasing the number of detected errors and reducing the number of false positives. Therefore, although the choice of thresholds influences the quality of the results, we give hints on how to choose *good* values.

5.4 Threats to the Validity of the Results

In this section we analyse the threats to the validity of the results obtained from our experiments. We consider three different categories.

- Threats to *internal validity* consider uncontrolled factors that might be responsible for the obtained results.
- Threats to *external validity* consider whether the conclusions of our experiments can be generalised and adapted to other situations.
- Threats to *construct validity* consider whether we are working with properties of interest.

Our first concern regarding threats to internal validity was the presence of faults in the code. Fortunately, we had the advantage of building on top of our previous work. Therefore, we could reuse code to prepare and process maps and to apply MRs to maps. Still, the MRs presented in this paper are new and include a novel component (thresholds). Therefore, we thoroughly tested the implementation of our MRs, using carefully constructed examples for which we could (manually) check the results. In particular, we were specially careful with the main component that increases the automation of our framework, the generation of follow-up inputs, and with the novel feature of our framework, the use of thresholds. We meticulously checked that follow-up inputs were uninterruptedly produced and there were no repetitions. Concerning thresholds, in addition to use values that allowed us to manually check that they were behaving as expected, we *metamorphically tested* all of them. For example, if we use a higher value of δ in `TagsCOMPkv`, then this MR should identify less potential errors. The choice of a specific set of MRs also falls in this category. The usefulness of our framework depends on its capability to find incorrect tags and detect the absence of tags in certain circumstances. This is strongly related to having *good* MRs. We carefully defined our MRs and discussed them with colleagues also working on MT.

Concerning threats to external validity, we first analysed whether our MRs were stable, regarding their capability to find errors, in different situations. In order to increase heterogeneity, we studied areas in four representative cities of OSM located in three different continents. We think that if our MRs are able to provide good results in so different *case studies*, then we can diminish the impact of good/bad contributors and of different languages. Second, we considered whether our framework can analyse other formats. We are confident that if a system would allow us to access its data, then we should be able to successfully apply our framework. Unfortunately, commercial applications such as Bing, Google, Wikimapia and Yandex do not provide access to their data and, therefore, we cannot appropriately address this threat.

Finally, in order to assess the impact of the potential threats to construct validity, we evaluated both the error detection ability of the seeds (source inputs) and the performance of the MRs in different areas. In order to diminish this threat, we used areas of different sizes, ranging from small to very big (see Fig. 1). Undetected errors in the definition of our proposed MRs, in the generation of follow-up inputs and in the results obtained by thresholds are also a threat to construct validity. We control this threat by executing a wide range of experiments that were able to completely cover the analyzed areas with the exception of two big areas for three MRs. Since we use XQuery, and the compliant XQuery 3.1 processor BaseX, there is a clear dependence of our framework on external programs. However, XQuery is a standardized query language[17] recommended by the World Wide Web Consortium (W3C), with many practitioners and wide acceptance in the industry.

Finally, all the tools and (areas of) maps used in this paper are freely available at https://github.com/jalmenUAL/MT-OSM. We hope that this helps reducing the potential threats since any user of our framework can independently check, replicate and analyse our findings.

6 Conclusions and Future Work

We have presented an MT approach to validate the information included in OSM maps. Specifically, we were interested in the consistency of the information included in the tags belonging to the elements of the map and in missing information and erroneous definitions of geometries. Taking as starting point our previous work on OSM, we defined MRs that should be able to find errors in maps. The experimental evidence obtained with our experiments clearly shows the efficacy of our framework. Actually, we have found errors with all the MRs and in all the cities that we analysed. In addition, many of these errors have not been previously reported in *Osmose*, the standard framework to report errors in OSM. Our framework is fully implemented, it has a high level of automation and can be used to find errors in any OSM map.

Our approach provides a generalisation of MT in which MRs include thresholds playing the role of tolerance level to which outputs are considered similar. The use of similarity degrees can be useful for other MR testers. For instance, MT has been applied to machine translation services in which a certain degree of similarity is required to compare translations [46]. We think that our *statistics-based* approach to deal with MRs that are not necessary properties can be successfully used in other application fields where imprecision in the definition of the MRs is inherent. For example, there are situations where we know that the result must be smaller than a certain value but we cannot compute this exact value. Another example is to deal with properties such as "almost all elements of a certain set must fulfil P". Finally, we think that our methodology, with the proper adaptations, can be used in other areas where tags play a fundamental role. Let us emphasise that in the case of OSM, we test a map by analysing the information that we extract from the map. It is useful that we have direct access to the data and we take advantage of this. However, we are more ambitious and think that we can go one step forward. Consider another mapping system where we do not have direct access to the data of the map (e.g. google maps), that is, the map is a black box: we ask questions (inputs) and receive a list of solutions (outputs). We can use our MRs to test this black-box. Obviously, we will have to (strongly) change our implementation because we will have to interact with a certain API, but the formal framework remains the same. Therefore, our final goal is to test maps by applying inputs, observing outputs and deciding (in our case, using MT) whether these observations are the expected ones.

We have several lines for future work. First, we will try and extend our approach to other systems. The labelling of the textual and multimedia information by end users is present in systems such as Flickr and Google images, being crucial for more precise searches and indexing. In addtion, it is one of the basis of media systems such as Twitter, Instagram and Youtube in which the hashtag labelling mechanism became very popular. Although some restrictions, guidelines and recommendations should be given for labelling to be useful for other users and search systems, the automatic detection of wrong labelling, from a comparison of similar objects, will improve the labelling of such systems. Our approach pursues this goal for the case of OpenStreetMap, in which key-value pairs act as tagging systems for geographic objects, and we will investigate whether it can be extended to deal with finding errors in tags included in social networks. We are also working on a user-friendly implementation of our framework. Although all the MRs are fully implemented and can be applied to analyse any OSM map provided by a user, we think that a GUI would increase the usability of our framework, in particular, among users that are not proficient in programming. An interesting feature of this GUI would be to offer the option of modifying the map by including the detected errors. One more difficult step forward is to implement a *recommender system*. The system would not only compute a list of potential errors; it would also provide a numerical assessment on how likely is that a certain potential error is indeed a real error.

References

[1] P. Ammann and J. Offutt, *Introduction to Software Testing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[2] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Dept. Comput. Sci., Hong Kong Univ. Sci. Technol., Hong Kong, Tech. Rep. HKUST-CS98-01, 1998.

17. https://www.w3.org/XML/Query/

[3] T. Y. Chen, T. H. Tse, and Z. Q. Zhou, “Fault-based testing without the need of oracles,” *Inf. Softw. Technol.*, vol. 45, no. 1, pp. 1–9, 2003.

[4] S. Segura, D. Towey, Z. Q. Zhou, and T. Y. Chen, “Metamorphic testing: Testing the untestable,” *IEEE Softw.*, vol. 37, no. 3, pp. 46–53, May/Jun. 2020.

[5] H. Liu, F.-C. Kuo, D. Towey, and T. Y. Chen, “How effectively does metamorphic testing alleviate the oracle problem,” *IEEE Trans. Softw. Eng.*, vol. 40, no. 1, pp. 4–22, Jan. 2014.

[6] T. Y. Chen *et al.*, “Metamorphic testing: A review of challenges and opportunities,” *ACM Comput. Surv.*, vol. 51, no. 1, pp. 4:1–4:27, 2018.

[7] J. Bennett, *OpenStreetMap*. Birmingham U.K.: Packt Pub. Ltd., 2010.

[8] J. Jokar Arsanjani, A. Zipf, P. Mooney, and M. Helbich, Eds., *OpenStreetMap in GIScience: Experiences, Research and Applications* (Lecture Notes in Geoinformation and Cartography). Berlin, Germany: Springer, 2015.

[9] J. M. Almendros-Jiménez, A. Becerra-Terón, M. G. Merayo, and M. Núñez, “Metamorphic testing of OpenStreetMap,” *Inf. Softw. Technol.*, vol. 138, 2021, Art. no. 106631.

[10] F.-B. Mocnik, A. Zipf, and M. Raifer, “The OpenStreetMap folksonomy and its evolution,” *Geo-spatial Inf. Sci.*, vol. 20, no. 3, pp. 219–230, 2017.

[11] S. Segura, G. Fraser, A. B. Sánchez, and A. Ruiz-Cortés, “A survey on metamorphic testing,” *IEEE Trans. Softw. Eng.*, vol. 42, no. 9, pp. 805–824, Sep. 2016.

[12] Z. Q. Zhou, T. H. Tse, and M. Witheridge, “Metamorphic robustness testing: Exposing hidden defects in citation statistics and journal impact factors,” *IEEE Trans. Softw. Eng.*, vol. 47, no. 6, pp. 1164–1183, Jun. 2021.

[13] M. Zhang, J. W. Keung, T. Y. Chen, and Y. Xiao, “Validating class integration test order generation systems with metamorphic testing,” *Inf. Softw. Technol.*, vol. 132, 2021, Art. no. 106507.

[14] A. Núñez, P. C. Cañizares, M. Núñez, and R. M. Hierons, “TEA-Cloud: A formal framework for testing cloud computing systems,” *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 261–284, Mar. 2021.

[15] C. Lidbury, A. Lascu, N. Chong, and A. F. Donaldson, “Many-core compiler fuzzing,” in *Proc. 36th ACM SIGPLAN Conf. Program. Lang. Des. Implementation*, 2015, pp. 65–76.

[16] T. Y. Chen *et al.*, “Metamorphic testing for cybersecurity,” *Computer*, vol. 49, no. 6, pp. 48–55, 2016.

[17] S. Pugh, M. S. Raunak, D. R. Kuhn, and R. Kacker, “Systematic testing of post-quantum cryptographic implementations using metamorphic testing,” in *Proc. IEEE 4th Int. Workshop Metamorphic Testing*, 2011, pp. 2–8.

[18] M. Jiang, T. Y. Chen, Z. Q. Zhou, and Z. Ding, “Input test suites for program repair: A novel construction method based on metamorphic relations,” *IEEE Trans. Rel.*, vol. 70, no. 1, pp. 285–303, Mar. 2021.

[19] S. Segura, J. A. Parejo, J. Troya, and A. Ruiz-Cortés, “Metamorphic testing of RESTful web APIs,” *IEEE Trans. Softw. Eng.*, vol. 44, no. 11, pp. 1083–1099, Nov. 2018.

[20] Z. Q. Zhou, S. Xiang, and T. Y. Chen, “Metamorphic testing for software quality assessment: A study of search engines,” *IEEE Trans. Softw. Eng.*, vol. 42, no. 3, pp. 264–284, Mar. 2016.

[21] M. Olsen and M. Raunak, “Increasing validity of simulation models through metamorphic testing,” *IEEE Trans. Rel.*, vol. 68, no. 1, pp. 91–108, Mar. 2019.

[22] B. Yan, B. Yecies, and Z. Q. Zhou, “Metamorphic relations for data validation: A case study of translated text messages,” in *Proc. IEEE/ACM 4th Int. Workshop Metamorphic Testing*, 2019, pp. 70–75.

[23] Z. Q. Zhou, L. Sun, T. Y. Chen, and D. Towey, “Metamorphic relations for enhancing system understanding and use,” *IEEE Trans. Softw. Eng.*, vol. 46, no. 10, pp. 1120–1154, Oct. 2020.

[24] X. He, X. Wang, J. Shi, and Y. Liu, “Testing high-performance numerical simulation programs: Experience, lessons learned, and open issues,” in *Proc. 29th ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2020, pp. 502–515.

[25] J. Brown, Z. Q. Zhou, and Y.-W. Chow, “Metamorphic testing of navigation software: A pilot study with google maps,” in *Proc. 51st Hawaii Int. Conf. System Sci.*, 2018, pp. 1–10.

[26] J. Brown, Z. Q. Zhou, and Y.-W. Chow, “Metamorphic testing of mapping software,” in *Proc. Towards Integrated Web, Mobile, IoT Technol.*, 2019, pp. 1–20.

[27] Z. Hui, S. Huang, C. Chua, and T. Y. Chen, “Semiautomated metamorphic testing approach for geographic information systems: An empirical study,” *IEEE Trans. Rel.*, vol. 69, no. 2, pp. 657–673, Jun. 2020.

[28] L. C. Degrossi, J. Porto de Albuquerque, R. d. Santos Rocha, and A. Zipf, “A taxonomy of quality assessment methods for volunteered and crowdsourced geographic information,” *Trans. GIS*, vol. 22, no. 2, pp. 542–560, 2018.

[29] A. Basiri, M. Haklay, G. Foody, and P. Mooney, “Crowdsourced geospatial data quality: Challenges and future directions,” *Int. J. Geographical Inf. Sci.*, vol. 33, no. 8, pp. 1588–1593, 2019.

[30] H. Zhang and J. Malczewski, “Quality evaluation of volunteered geographic information: The case of OpenStreetMap,” in *Proc. Crowdsourcing: Concepts, Methodologies, Tools, Appl.*, 2019, pp. 1173–1201.

[31] S. S. Sehra, J. Singh, H. S. Rai, and S. S. Anand, “Extending processing toolbox for assessing the logical consistency of OpenStreetMap data,” *Trans. GIS*, vol. 24, no. 1, pp. 44–71, 2020.

[32] M. Haklay, S. Basiouka, V. Antoniou, and A. Ather, “How many volunteers does it take to map an area well? The validity of linus law to volunteered geographic information,” *Cartograph. J.*, vol. 47, no. 4, pp. 315–322, 2010.

[33] P. Mooney and P. Corcoran, “Characteristics of heavily edited objects in OpenStreetMap,” *Future Internet*, vol. 4, no. 1, pp. 285–305, 2012.

[34] C. Keßler and R. T. A. De Groot, “Trust as a proxy measure for the quality of volunteered geographic information in the case of OpenStreetMap” in *Proc. Geograph. Inf. Sci. Heart Euro.*, 2013, pp. 21–37.

[35] C. Barron, P. Neis, and A. Zipf, “A comprehensive framework for intrinsic OpenStreetMap quality analysis,” *Trans. GIS*, vol. 18, no. 6, pp. 877–895, 2014.

[36] A. Vandecasteele and R. Devillers, “Improving volunteered geographic information quality using a tag recommender system: The case of OpenStreetMap,” in *Proc. Openstreetmap Giscience.*, 2015, pp. 59–80.

[37] S. S. Sehra, J. Singh, and H. S. Rai, “Assessing OpenStreetMap data using intrinsic quality indicators: An extension to the QGIS processing toolbox,” *Future Internet*, vol. 9, no. 2, 2017, Art. no. 15.

[38] M. Minghini and F. Frassinelli, “OpenStreetMap history for intrinsic quality assessment: Is OSM up-to-date?,” *Open Geospatial Data, Softw. Standards*, vol. 4, no. 1, pp. 9:1–9:17, 2019.

[39] J. M. Almendros-Jiménez and A. Becerra-Terón, “Analyzing the tagging quality of the spanish OpenStreetMap,” *ISPRS Int. J. Geo-Inf.*, vol. 7, no. 8, pp. 323:1–323:26, 2018.

[40] J. M. Almendros-Jiménez, A. Becerra-Terón, and M. Torres, “Integrating and querying OpenStreetMap and linked geo open data,” *Comput. J.*, vol. 62, no. 3, pp. 321–345, 2019.

[41] J. M. Almendros-Jiménez, A. Becerra-Terón, and M. Torres, “The retrieval of social network data for points-of-interest in OpenStreetMap,” *Hum.-Centric Comput. Inf. Sci.*, vol. 11, no. 10, pp. 1–22, 2021.

[42] A. L. Ali, N. Sirilertworakul, A. Zipf, and A. Mobasheri, “Guided classification system for conceptual overlapping classes in OpenStreetMap,” *ISPRS Int. J. Geo- Inf.*, vol. 5, no. 6, 2016, Art. no. 87.

[43] P. Neis, M. Goetz, and A. Zipf, “Towards automatic vandalism detection in openstreetmap,” *ISPRS Int. J. Geo- Inf.*, vol. 1, no. 3, pp. 315–332, 2012.

[44] R. Bamford *et al.*, “XQuery reloaded,” *Proc. VLDB Endowment*, vol. 2, no. 2, pp. 1342–1353, 2009.

[45] J. Robie, D. Chamberlin, M. Dyck, and J. Snelson, “XQuery 3.0: An XML query language,” W3C Proposed Recommendation, World Wide Web Consortium (W3C), Cambridge, MA, USA, Tech. Rep., 2014. [Online]. Available: https://www.w3.org/TR/xquery-30/

[46] D. Pesu, Z. Q. Zhou, J. Zhen, and D. Towey, “A monte carlo method for metamorphic testing of machine translation services,” in *Proc. IEEE/ACM 3rd Int. Workshop Metamorphic Testing*, 2018, pp. 38–45.