Combining Metamorphic Testing and Machine Learning to Enhance OpenStreetMap

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

***Abstract*—Metamorphic testing (MT) is a useful tool to test systems where an oracle is not available. MT relies on the definition of metamorphic relations (MR), that is, certain properties that relate a set of inputs and the set of outputs produced by the system under test (SUT) as response to these inputs. Usually, a violation of an MR implies that the SUT is faulty. However, some work on MT accepts, for certain SUTs, that the violation of an MR *almost always* is a symptom of an error but assumes the potential existence of false positives. This is the case, for instance, of our recent work where we applied MT to improve OpenStreetMap (OSM). Our MRs were able to uncover a large amount of errors in all the analyzed maps but we suffered the presence of a nonnegligible number of false positives. Therefore, an expert had to manually check the suspicious elements identified by our MRs. If we analyze large maps, then this manual task is unfeasible. In this article we solve the main limitation of our previous approach: we accurately and automatically discard false positives. Our new framework combines MT, along the same lines of our previous work, and machine learning. Specifically, we provide three models, one for each MR, based on the random forest model. The models were extensively trained with real data obtained from the application of MRs to maps of cities located in different continents. In order to evaluate the usefulness of our models, we tested them using different cities, in countries that were not considered in the training set. The results were very good: accuracy of the models is never lower than 0.90, it is usually much higher and in many situations reaches 1.0. The computation of the F1-scores yielded similar results.**

***Index Terms*—Machine learning (ML), metamorphic testing (MT), openstreetmap (OSM), quality of maps.**

I. Introduction

IN CLASSICAL software testing [1], [2], the tester selects a set of inputs, applies them to the system under test (SUT), and decides whether the obtained outputs correspond to valid behaviors of the system. This decision can rely on an *oracle* that will be in charge of providing a verdict about the correctness of the observed behaviors. Unfortunately, oracles are not always available [3] and, in this case, the testing process mainly relies on the experience and knowledge of the tester about the expected behavior of the SUT. This process is mainly manual, time-consuming, and cannot be applied to complex systems. Moreover, in some situations even an expert tester would be unable to provide a verdict. For example, consider that a program P is an implementation of the sine function. If we are going to test this program with an input such as 43, what is the expected result of $P(43)$?

Metamorphic testing [4], [5], [6] (MT) is a very useful testing technique to analyze systems where the tester does not have an oracle. The distinguishing characteristic of MT with respect to a classical testing approach is that while in the latter we isolatedly analyze the observed output after the application of a given input, in MT we jointly analyze the answers to several inputs. Then, we check whether certain expected properties, called *metamorphic relations* (MR), between the applied inputs and the observed outputs hold. Coming back to our simple example, we might not know the value of $\sin(43)$ but we do know that if either $P(43) \neq P(403)$ or $P(43) \neq -P(223)$, then P is faulty. In the previous presentation, there are two different types of inputs. *Source inputs* are provided by the tester and should cover the main features of the system. *Follow-up inputs* are used during the iteration of the process. An MT framework should be able to provide a method to, as much as possible, automatically generate these inputs. In our running example, 0 would be a good source input and 360 or -360 would be follow-up inputs.

OpenStreetMap (OSM) is a volunteered geographic information [7] system in which nonprofessional users contribute to the construction of a world map. Essentially, a map is a set of *nodes*. Some of these nodes can be grouped to define *ways*. The term *way* is used in a broad sense: it can be a highway or a street but it can also represent the perimeter of a building or a river. The textual information describes the attributes of the map objects such as the name, the type of object (for instance, *highway*, *building*, and *amenity*), and any other relevant information about the object. The textual information is represented by (key, value) pairs called *tags*. In previous work, taking as initial step our work on OSM [8], [9], [10], we defined and implemented an MT framework [11] to analyze OSM maps. Our experiments in different cities showed that OSM maps are plagued with wrong and inaccurate information concerning the *geometries* included in OSM. Examples of these problems were roundabouts without

Manuscript received 30 September 2022; revised 1 May 2023, 24 October 2023, and 17 January 2024; accepted 12 March 2024. Date of publication 4 April 2024; date of current version 3 December 2024. This work was supported by the State Research Agency (AEI) of the Spanish Ministry of Science and Innovation, under Grant PID2019-104735RB-C42 (SAFER) and Grant PID2021-122215NB-C31 (AwESOMe). Associate Editor: H. Jiang. *(Corresponding author: Manuel Núñez.)*

Manuel Méndez, Mercedes G. Merayo, and Manuel Núñez are with the Design and Testing of Reliable Systems research group, Universidad Complutense de Madrid, 28040 Madrid, Spain (e-mail: manumend@ucm.es; mgmerayo@fdi.ucm.es; mn@sip.ucm.es).

Antonio Becerra-Terón and Jesús M. Almendros-Jiménez are with the Department of Informatics, Carretera de Sacramento s/n, University of Almería, 04120 Almería, Spain (e-mail: abecerra@ual.es; jalmen@ual.es).

Data is available online at https://github.com/jalmenUAL/MT-OSM.

Digital Object Identifier 10.1109/TR.2024.3379366

exit and highways that did not intersect with other highways. We went one step further in the analysis of maps and dealt with the information included in the nodes of the map. We presented an MT framework [12] where our goal was to reveal tagging mistakes in OSM. Typically, simple tagging mistakes are found by contributors because they usually review each other, but our previous work showed that this is not enough. For example, it is relatively easy to detect whether a contributor uses an unknown and, therefore, unallowed label, even if the number of allowed OSM labels is huge. However, it is more difficult to detect situations where a contributor incorrectly uses an allowed label (e.g., it is wrongly combined with other labels) because the identification of this misuse requires *guessing* the type of object that the contributor desires to describe and the OSM rules that need to be applied. A first attempt is to report *suspicious* labels but we will have labels that are rarely used but allowed, producing false positives. We considered a more sophisticated strategy: our MRs aimed at detecting *agreement patterns*. The rationale was that if a certain pattern is repeated very often, then it is very likely that a deviation from this pattern violates the (implicit) tagging rules. However, this deviation could represent a valid piece of information and we would end up, again, with a false positive. Our MT framework included a novel feature to characterize a contributors' agreement: equip MRs with thresholds. Intuitively, an agreement will appear when we have a *large* number of occurrences of a certain pattern and a *low* number of discrepancies. Unfortunately, the percentage of false positives is relatively high and increases with the number of elements in the considered map. For example, the application of our MRs to different parts of Nairobi (Parkroad, Landingtong, and Hurlingham) and London (Chelsea, ConventGarden, and Notting Hill) returned, respectively, around 19% and 28% false positives.[1] Discarding false positives is extremely time-consuming, as our OSM expert needed an average of one week per city to discard false positives resulting from the application of our MRs. In fact, some of the (potential) errors were so subtle that only a real expert on OSM would be able to decide whether it was a false positive or not. In this article we present a new MT framework that solves the issue by training supervised machine learning (ML) models.

We had to choose the most suitable ML model to implement our framework. First, we took into account the exclusive presence of categorical variables (in a large number). Thus, we discarded regression techniques (e.g., linear regression or support vector machine) because of the absence of continuous variables. We also discarded deep learning methods (e.g., multilayer perceptron neural networks or convolutional neural networks) because of their difficulty to perform a strict classification, as we require. The obvious choice was between two algorithms: *random forest* (RF) and *classification tree*. A RF classifier model is an ensemble supervised ML model that contains a defined number of classification trees. The most voted-for output of the classification trees is the final output of the RF. We initially thought that classification trees would be enough but our first experiments showed that, in order to avoid the possible problems derived from a human failure in the labeling, it would be better to use a RF. In our context, the detection of false positives is crucial to improve the accuracy results. With this aim, other ML-based solutions could have been proposed. However, for each MR we need to train and test different keys and values, such as *building*, *highway*, *amenity,* and *tourism*, among others. Actually, we need a global classifier being able to mix different classifiers trained for each analyzed tag. Precisely, it is in this context where RF is the best ML-based solution.

In order to *train* our models, one per MR, we used as training set the results of the application of our MRs in eleven cities around the world. Our goal was to select cities that significantly differed both in the location and in the number of total elements included in the map. By continent, the distribution is as follows: six cities in Europe (Almeria, Banja Luka, London, Madrid, Paris, and Valencia) and one city in North America (New York), Central America (City of Mexico), South America (Sao Paolo), Africa (Nairobi), and Asia (Tokyo). The chosen cities can be also categorized based on the number of elements included in the map, with some cities having a low number (e.g., Almeria and Banja Luka), some having a medium number (e.g., Nairobi and Valencia), and others having a high number (e.g., New York and Tokyo).

The second phase was to *test* our models: we evaluated them in ten cities. As in the case of the training set, these cities significantly differed both in the location and in the number of total elements: three cities in Europe (Barcelona, Brussels, and Rome), two cities in North America (Boston and Ottawa) and Africa (Cairo and Luanda), and one city in Central America (La Havana), South America (Santiago), and Asia (Wuhan). By number of elements, there are cities with a low number (e.g., Cairo and Luanda), a medium number (e.g., Wuhan and Ottawa), and a high number (e.g., Barcelona and Rome). In order to evaluate the quality of the results, we used two representative metrics that target different characteristics of the model: *accuracy* (computing the percentage of predictions that our model got right) and *F1-score* (combining the *precision* and the *recall* of the model in a single value).

The results obtained by our models were very good. For all the cities and MRs we obtained accuracies higher than 0.9, reaching in several situations 1.0. These results strongly improve the ones obtained if we simply use our MRs [12]. In that case, we ran different experiments for four cities (i.e., Madrid, London, Sao Paolo, and Nairobi) and, even after carefully choosing thresholds to reduce the percentage of false positives, we obtained worse accuracies for the MRs. The values of the F1-scores are slightly lower but show a similar trend: most values are higher than 0.9, reaching a value of 1.0 for the same experiments where the accuracy is equal to 1.0. It is worth mentioning that our combination of MT and ML also outperforms the application of a single ML model. We performed an experiment by training with data obtained from the city of Madrid and testing the model in Barcelona and Almeria, two Spanish cities. In this case, we considered *all* the nodes of the map, in contrast with using MRs where only *potentially wrong* nodes are selected. The results showed that the accuracy obtained by applying only

[1] Nairobi and London are typical examples of cities with, respectively, a low/medium and a high number of elements.

ML was 0.836 whereas the application of our approach was able to obtain an accuracy of 0.925. As a concluding remark, the combination of MT and ML makes possible, with little effort, to detect most of the tagging errors in OSM maps while discarding most false positives obtained by the application of MRs. Certainly, we needed to extensively train our ML models with elements manually labeled as true/false positives by an OSM expert. However, after the models were trained, they can receive a set of potential errors and will automatically discard false positives, with a high accuracy, so that the OSM expert is not needed.

The rest of the article is organized as follows. In Section II, we review work related to the topics of the article. In Section III, we give a brief introduction to OSM. In Section IV, we present our modeling of OSM, our previous MT framework, and the formal definition of our MRs. In Section V, we present the main features of RFs and give some implementation details concerning the specific models that we associate with each MR. In Section VI, we present our new framework where we combine MT and ML to analyze OSM. In Section VII, we present the results of the experiments that support the validity of our framework. We pose some research questions, answer them, and analyze the threats to the validity of the obtained results. Finally, Section VIII concludes this article and presents some lines of future work.

II. Related Work

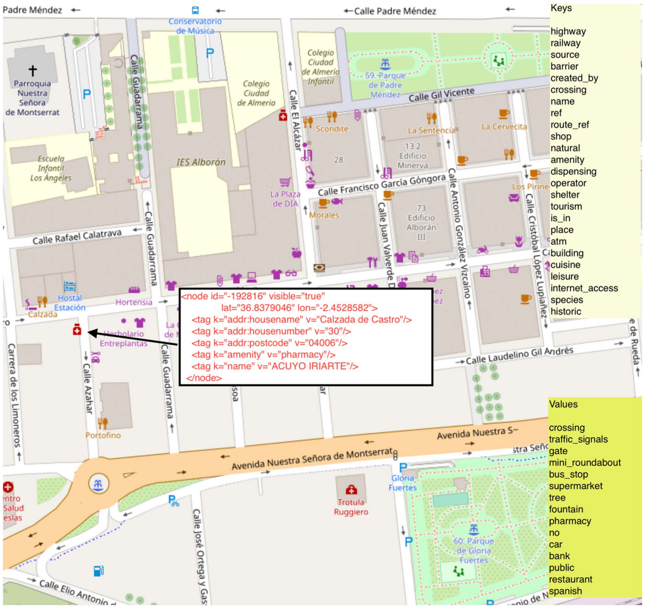
In this section, we briefly review the most related work to the topic of this article. First, MT is a highly versatile technique that has been used in many heterogeneous and unrelated fields [13], [14]. To mention a few, and with the goal of showing its versatility, MT has been used in citation database systems [15], cloud [16], cryptography [17], image recognition systems [18], program repair [19], quantum computing [20], search engines [21], and simulation [22].

More related to the contents of this article, we can mention the notion of *metamorphic exploration* [23]. The idea is that an MR sometimes finds situations that do not represent an error of the SUT but that may facilitate the understanding of a behavior of the system. Several approaches [24], [25], [26], [27] have noticed that even well designed MRs can produce false positives too. This is the case when the SUT behaves slightly different, but still correctly, to the response of the MR when small changes are made to the inputs (for instance, using harmless mutations). For systems in which output is complex enough (compilers, translators, and debuggers, among others), two outputs might not be syntactically equal but be semantically equivalent. Nonfunctional properties such as execution time, memory consumption, and energy usage are inherently nondeterministic and, thus, can produce false positives too [26]. The same happens with numeric values [25], where in certain cases slight variations in the outputs (for instance, floating point calculations) are not actually real errors. Authors dealing with false positives have been forced to manually check them [27] or to impose certain thresholds [26], [28]. In addition, some MRs have to assume a certain error margin. This is the case, for example, for translation services (Google, Bing, etc.) [29]. The output of a translation service is a string (a sentence) in a given language. For two similar inputs, the MR should take into account a certain similarity between outputs to avoid false positives. In fact, the authors of this work used well-known metrics—Levenshtein distance and cosine similarity—to compare them. Another solution, the one that we explore in this article, is to combine MT with a statistical technique. Our goal is to reduce the number of false positives in our MRs by automatically discarding them by using an ML model associated with each MR. The key of the success of such a method is that false positives can be grouped and each group has the same cause. Thus, a pattern can be established to (automatically) recognize false positives.

There is not much work on the application of MT to analyze maps. In addition to our previous work [11], [12], we can mention the use of MT to validate maps [30], [31]. However, the main goal of our work and this one diverges because we analyze the information appearing in the map, specifically, the tags labeling the nodes, while this work considers distances and paths between points of the map. Another work uses MT to analyze geographic information systems [32]. In this case, the work assesses whether the computation of superficial areas is correct. Again, this approach and ours are unrelated.

ML has been frequently applied in OSM (see [33] for a survey). Some of the approaches focus on learning properties from geometric features. It is possible to infer the value for the tag *lanes*, which specify how many traffic lanes a road has, by analyzing the polygons formed by the road network [34], [35]. This is very useful because the *lanes* field is usually empty in OSM and, thanks to the use of ML, such a property can be filled. ML has also being used to train a recommended system. For example, the OSMRec tool [36] uses geometrical and textual features to train an support vector machine (SVM) classifier for recommending tags for new objects. Trained classifiers can also find possible annotation errors. For example, a K-nearest neighbors (KNN) classifier has been used to analyze tags associated with four types of green areas from geometrical, topological, and contextual features [37]. Our proposal has also as goal to find possible annotation errors, but we combine MT with ML, so that our ML models can focus on the most problematic tags. Some other issues handled by ML-based techniques are vandalism detection [38], socio-economic indicators inference [39], experienced contributors detection [40], and roads speed estimation [41].

Finally, since we combine MT and ML, we will review some work where these two approaches are simultaneously used. Given the difficulty to test ML models, it was only natural that MT approaches would eventually appear. In this case, we can mention the use of MT to analyze classifiers [42], [43], [44], [45], forecasters [46], unsupervised ML models [47], the training process of neural networks [48] and linear regression systems [49]. A completely different line of work uses ML to enhance MT, in particular, to find MRs [50], [51], [52]. To the best of the authors' knowledge, there are no previous MT frameworks where ML is used to improve the results of the testing process.



```
<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. OSM of Almeria (excerpt).

III. Introduction to OSM

OSM is a free, open-source alternative to commercial mapping services. According to OSM statistics, [2] OSM has 8.3 million registered users, 7.4 billion nodes, approximately 4 million map changes/day, and 1.75 million different user contributors. Typically, anonymous (but registered) users edit OSM maps and complete the spatial and textual information of the map. In Fig. 1, we can find an excerpt of the OSM corresponding to the city of Almeria (Spain). The boxed item includes the information associated with a certain node: unique identifier, latitude and longitude, and a set of (key, value) pairs, its set of tags. In this case, we can see that the node identifies a pharmacy [see the (*amenity*, *pharmacy*) tag] located at *Calzada de Castro* street, 30. The name of the pharmacy is *ACUYO IRIARTE*.

The OSM community has established certain rules that must be followed for both spatial and textual information. For instance, buildings never overlap and pharmacy is a subtype of amenity, which means that amenity is a key and pharmacy is a value in such a way that (*amenity*, *pharmacy*) is mandatory for pharmacies. Some rules are mandatory, whereas others are recommended and optional. For instance, the *dispensing* key—whether a pharmacy dispenses prescription drugs or not—with a yes/no value is recommended but not mandatory. Thus, in some pharmacies the (key, value) pair corresponding to dispensing can be missing. OSM contributors should stick to a certain set of rules to add information to a map, in particular, concerning both mandatory and recommended combinations of (key, value) pairs and combinations of the keys that should appear in an object. This has produced a so-called *folksonomy* [53], the tags that should be used to thematically describe OSM elements, that has been collected in several websites, such as *Taginfo* and *Tagfinder*.[3] The OSM website also publishes all the rules for each specific type of object. For instance, we can find the rules for pharmacies[4] and, in particular, the most known error for pharmacy: the use of the (*shop*, *pharmacy*) pair instead of (*amenity*, *pharmacy*). OSM contributors are responsible for reviewing the contributions of their peers in such a way that they can both report and correct what other users do wrong. Nonetheless, manual checking and repair is costly, which has led to a growing interest in developing automated tools.[5] These tools are able to report bugs and errors in OSM maps and assist, monitor, and visualize OSM contributions. Among them, we can mention: *iOSMAnalyzer* tool,[6] *Quantum GIS (QGIS)*,[7] *OSMantic*,[8] *Keep Right*,[9] and *OSM inspector*.[10] The most widely used is *Open Street Map Oversight Search Engine (Osmose)*.[11]

IV. Our MT Framework: MR

In this section we briefly present the main details about the MT framework that will be the basis to apply our ML models. In particular, we present the formal definition of our MRs. The interested reader is referred to our previous work [12] for longer explanations, examples, and implementation details. In particular, the MRs utilized in this article were first introduced in that particular work. First, we present a formal definition of a map. Intuitively, a map is a set of nodes (labeled by a set of tags containing the relevant information about the node) together with a set of ways (a sequence of nodes).

Definition 1: A *tag* is a pair (k, v), where k is a string associated with a key and v is a string associated with a value. Tags will provide information about nodes and ways. Examples of valid tags are *(railway, stop)*, *(highway,crossing)*, and *(building, yes)*. A *node* is a triple (id, (lat, long), Tg), where id is the unique identification of the node, lat and long are its latitude and longitude and Tg is its set of tags. For example, in Fig. 1 we can see a node representing a pharmacy in Almeria (Spain). A *way* is also a triple (id, $\langle n_1, \dots, n_s \rangle$, Tg), where id and Tg are as before, and $\langle n_1, \dots, n_s \rangle$ is the sequence defining the way. Note that, in general, a way represents a polyline and if $n_1 = n_s$, then it represents a polygon.

Let e be either a node or a way. We have that tags(e), keys(e), and vals(e) are, respectively, the set of tags of e and the keys/values appearing in tags(e). We generalize these notions to set of elements E in the expected way: 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).

Given a set of nodes $\mathcal{N}$, let $W_{\mathcal{N}}$ be ways such that all their nodes belong to the set $\mathcal{N}$. An *OSM map* is a pair $(\mathcal{N}, \mathcal{W}_{\mathcal{N}})$. Abusing the notation, if m is a map then we will write $e \in m$

[2] [Online]. Available: https://wiki.openstreetmap.org/wiki/Stats

[3] [Online]. Available: http://taginfo.openstreetmap.org and http://tagfinder.herokuapp.com.

[4] [Online]. Available: https://wiki.openstreetmap.org/wiki/Tag:amenity=pharmacy

[5] [Online]. Available: https://wiki.openstreetmap.org/wiki/Quality_assurance

[6] [Online]. Available: https://github.com/zehpunktbarron/iOSMAnalyzer

[7] [Online]. Available: https://www.qgis.org/

[8] [Online]. Available: https://sites.google.com/site/osmanticjosm/

[9] [Online]. Available: https://keepright.at/

[10] [Online]. Available: https://tools.geofabrik.de/osmi/

[11] [Online]. Available: https://osmose.openstreetmap.fr/

if $e \in \mathcal{N} \cup \mathcal{W}_{\mathcal{N}}$. As usual in OSM, we assume that for each element of a map e and key k appearing in a tag of the map, there exists a unique v such that $(k, v) \in \text{tags}(e)$.

In order to define MRs, we need to introduce the concepts of *input* and *output*.

Definition 2: A pair $(T, (s_1, s_2))$ is an *input* if $T \in \{*, \text{node}, \text{way}\}$ and (s_1, s_2) is a pair of regular expressions (they may contain the $*$ wild character) representing patterns for keys and values, respectively. Examples of these pairs are *(name ,A7)* and *(highway,*)*.

The application of an input $i = (T, (s_1, s_2))$ to a map $\mathcal{M}$, denoted by $\mathcal{M}(i)$, includes all the elements of the map of type T having a tag matching (s_1, s_2). We say that $\mathcal{M}(i)$ is the *output* corresponding to i.

Let $\mathcal{M} = (\mathcal{N}, \mathcal{W}_{\mathcal{N}})$ be an OSM map and $\delta, \epsilon \in \mathbb{N}$ be thresholds. An MR $R \subseteq \mathcal{I}^2 \times \mathcal{P}(\mathcal{N} \cup \mathcal{W}_{\mathcal{N}})^2 \times \mathbb{N}^2$ is a relation over two inputs, their corresponding outputs, and the thresholds. We write $R(i_1, i_2, \mathcal{M}(i_1), \mathcal{M}(i_2), \delta, \epsilon)$ to denote $(i_1, i_2, \mathcal{M}(i_1), \mathcal{M}(i_2), \delta, \epsilon) \in R$.

Let δ and ϵ be thresholds, $i_1 = (T_1, C_1)$ and $i_2 = (T_2, C_2)$ be inputs, and $O_1 = \mathcal{M}(i_1)$ and $O_2 = \mathcal{M}(i_2)$ be the corresponding outputs. The definition of an MR R follows this pattern:

$$\boxed{\begin{array}{c} R(i_1, i_2, O_1, O_2, \delta, \epsilon) \\ \Updownarrow_{\mathbf{def}} \\ \mathcal{C}(\delta, \epsilon, T_1, C_1, T_2, C_2, O_1, O_2) \end{array}}$$

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

In the previous expression, we say that i_1 is a *source input* whereas i_2 is a *follow-up input*.

There is an important distinction between source and follow-up inputs. In our framework, while source inputs must be provided by testers, follow-up inputs are automatically generated from source inputs and the definition of the MR. Therefore, the whole MT process is highly automated because it is enough to provide a *seed* to initiate the process. In our previous work, we enumerated useful input seeds (in terms of the keys and values that should be used) that are able to cover large portions of maps in a small number of iterations.

In this article, we study three out of the four MRs previously defined: `TagsCOMPkvk`, `TagsCOMPkv`, and `TagsMISSkvk`. We have left `TagsName` aside because the targeted errors strongly depended on the semantics of the strings conforming tags and, therefore, it was not possible to create a learning model because the applications of this MR are too specific. For example, a typical application of the rule is that if an element has a tag matching *(name, *hotel)*, then it should also have another tag with *hotel* as value. Next, we present the formal definition of our three MRs. Explanations of the behavior of these MRs, examples of their use, how follow-up inputs are automatically generated from source inputs and appropriate values for the thresholds can be found in our previous work [12].

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

This MR checks that there are no elements in the studied map simultaneously having a certain tag and another tag with a key that it is not compatible with the first tag. For example, good OSM practices state that traffic signals do not have a name. Therefore, if we find a node n such that *(highway,traffic_signals)*$\in$ *tags(n)* and there exists a string β such that we also have $(\text{name}, \beta) \in \text{tags}(n)$, then we have the violation of an OSM rule.

Formally, given thresholds $\delta, \epsilon \in \mathbb{N}$, and strings α and β, let $i_1 = (*, (\alpha, \beta))$ and $O_1 = \mathcal{M}(i_1)$. Now, consider a string α' such that $\alpha' \in \text{keys}(O_1) \setminus \{\alpha\}$. Let $i_2 = (*, (\alpha', *))$ and $O_2 = \mathcal{M}(i_2)$. We have the following metamorphic relation:

$$\boxed{\begin{array}{c} \texttt{TagsCOMPkvk}(i_1, i_2, O_1, O_2, \delta, \epsilon) \\ \Updownarrow_{\mathbf{def}} \\ |O_1| \geq \delta \Longrightarrow |O_1 \cap O_2| \geq \epsilon. \end{array}}$$

In order to remind how thresholds work, we will informally explain their intended meaning in this MR. The idea is that if we have *many* occurrences of the (α, β) tag then we expect that *many* of the elements including this tag also include a tag having α' as key. Note that δ must be higher than ϵ.

B. `TagsMISSkvk`*: Missing Key If a (key,value) Appears in an Element*

Our second MR targets a completely different property: if a given tag appears in the set of tags of an element, then a certain key should also appear in that set. For example, OSM rules state that if we find the $(highway, crossing)$ tag in an element, then this element should also have a tag having *crossing* as key. We can also apply this rule to check whether recommendations are followed. For example, it is recommended by OSM good practices, although not mandatory, that an (amenity, restaurant) tag should be accompanied by a tag having *cuisine* as key. Given thresholds $\delta, \epsilon \in \mathbb{N}$, and strings α and β, let $i_1 = (*, (\alpha, \beta))$ and $O_1 = \mathcal{M}(i_1)$. Now consider another string $\alpha' \in \text{keys}(O_1) \setminus \{\alpha\}$ and let $i_2 = (*, (\alpha', *))$ and $O_2 = \mathcal{M}(i_2)$. We have the following metamorphic relation:

$$\boxed{\begin{array}{c} \texttt{TagsMISSkvk}(i_1, i_2, O_1, O_2, \delta, \epsilon) \\ \Updownarrow_{\mathbf{def}} \\ |O_1 \cap \overline{O_2}| \geq \delta \Longrightarrow |O_1 \cap O_2| \leq \epsilon \end{array}}$$

where $\overline{A}$ represents the complement of the set A.

C. `TagsCOMPkv`*: Compatibility Between Keys and Their Associated Values*

Our final MR ensures that certain combinations of keys and values are never paired together. This problem is strongly related to the fact that, ideally, values should be univocally associated with keys. For example, we should not have two elements such that one of them has a $(building, commercial)$ tag, whereas the other one has a $(landuse, commercial)$ tag. On the one hand, the *landuse* key is typically used to represent business areas and is, thus, associated with the *commercial* value. On the other hand, the *building* key is generally used to represent *public* or *residential* areas and should not be associated with the *commercial* value. In other words, for each possible value, there should be a *main* key associated with it. However, we have to be careful and exclude values, such as *yes*, *no*, and *numbers*, that can be naturally paired with several keys.

Given thresholds $\delta, \epsilon \in \mathbb{N}$, and a string α, associated with a key, let $i_1 = (*, (\alpha, *))$ and $O_1 = \mathcal{M}(i_1)$. Let $\beta \in \text{vals}(O_1)$ be a string and let $i_2 = (*, (*, \beta))$ and $O_2 = \mathcal{M}(i_2)$. We have the following metamorphic relation:

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

Our MRs have been implemented in XQuery [54], [55].

V. Our ML Models

In this section, we will present the fundamentals of the RF model (longer explanations can be found in the original work [56]). We will also describe the main characteristics and hyperparameters of our proposed models. Finally, we will give some details about the manual *overrides* that we have performed in the model associated with the `TagsCOMPkvk` MR.

A. RF Fundamentals

A RF is an ensemble ML method that is based on the combination of several *decision trees* with the same characteristics but trained with a randomly chosen portion of the training set. Thus, when the outputs of each tree are combined, the prediction generalizes much better than if we only use one tree. The most voted-for class will be the output class predicted by the RF classifier model.

A *decision tree* aims to develop a model that learns and predicts the value of a target variable by learning decision rules inferred from the predictor variables of the data. The tree structure begins with a single node, which represents all the observations, that branches into two possible outcomes. This process is repeated in each node until we obtain end nodes, the leaves of the tree that encode the output of each decision path. The algorithm used to split nodes in a tree of an RF relies on the *Gini index*. Although its main application is to measure economic inequality in a country [57], it has been also used in other fields such as the prediction of precipitation [58] and in the search of optimal models for the plan of total control of environmental pollutants [59]. This index is used in RFs to measure the impurity of a node, being the aim in each split to minimize the Gini index and achieve greater homogeneity in the resulting subsets. It can be formally defined as follows:

$$G = 1 - \sum_{i=1}^{K} P_i^2$$

where P_i is the proportion of observations in the node belonging to the ith class and K is the number of classes, that is, the possible outputs of the model. Values close to 0 denote that the node is close to perfect homogeneity, and values close to 1 denote that the node is close to total heterogeneity.

In each node, in order to determine what will be the next split, the algorithm considers all the possible splits, computes the average of the Gini index measured in the two generated child nodes, and chooses the one with the lowest value.

B. Our RF Model

In this section, we explain the general characteristics of our RF models and the similarities and differences between them. First, we have to explain that the input features of the RF models depend on the MR. For example, in the case of `TagsCOMPkvk`, the input features are the different combinations of (possibly conflicting) key values with respect to the analyzed (key, value) pair. For example, The *amenity* key generates a conflict with the (*building*, *school*) pair. In addition, the target binary variable encodes whether the input feature represents an error or a false positive.

We trained three RF models, one per MR, with the same hyperparameters. The hyperparameters were obtained after performing a fine-tuning, using a grid-search process, which consists of selecting the best combination after trying all possible combinations of considered hyperparameters. Next, we will explain the initially considered and the optimal values by hyperparameter.

For the *number of classification trees* and the *maximum depth of the tree*, values higher than 10 and 125, respectively, did not result in a significant improvement in the quality of the model. Therefore, we set those values in order to reduce the computational time as much as possible. For the *minimum samples leaf* and the *minimum samples split*, we experimented with typical values (from 1 to 10) and found the best values to be 1 and 4, respectively. Similar experimentation was conducted for *minimum impurity decrease* and the best value was found to be 10^{-6}. Finally, we found that the best class weights determined by the grid-search were *balanced* weights. We have two concluding remarks. First, we exercised around 50 000 combinations of hyperparameters, with the corresponding training and evaluation of the resulting models. Second, due to the high value of *maximum depth of the tree*, our model has significant capabilities. This can be observed by considering that the number of decision trees participating in our RFs have, on average, 240 final nodes.

In order to avoid errors due to underfitting, we have to perform manual improvements to the model associated with the `TagsCOMPkvk` MR, that is, we preset the answer of the model for certain relevant combinations of source and follow-up inputs that were not included in the training set but that follow the same

TABLE I
TRAINING CITIES

City (Country)	*building*	*highway*	*amenity*	*tourism*
Europe				
Almeria (Spain)	5901	4959	1486	259
Banja Luka (Bosnia)	11 492	5254	812	129
London (UK)	22 635	11 144	3492	284
Madrid (Spain)	20 125	9947	3944	378
Paris (France)	21 075	10 475	4167	269
Valencia (Spain)	15 988	7483	2128	176
North America				
New York (USA)	24 865	11 360	4389	368
Central America				
City of Mexico (Mexico)	25 836	10 027	3220	343
South America				
Sao Paolo (Brazil)	26 016	9641	1648	259
Africa				
Nairobi (Kenya)	22 025	7209	2052	240
Asia				
Tokyo (Japan)	24 321	12 276	4125	414

Distribution of analyzed tags.

TABLE II
APPLICATION OF `TagsCOMPkvk`

City	*building*	*highway*	*amenity*	*tourism*	*OSM Expert*
Almeria	106	96	89	30	321
Banja Luka	115	124	79	17	335
London	280	200	334	41	855
Madrid	212	152	324	45	733
Paris	232	303	314	37	886
Valencia	281	279	224	28	810
City of Mexico	299	250	384	45	978
New York	279	289	455	55	1078
Sao Paolo	304	193	433	60	990
Nairobi	218	130	135	28	511
Tokyo	261	215	287	30	793
Total	**2587**	**2231**	**3058**	**414**	**8290**

Tags require labeling by an OSM expert on a city-by-city basis.

TABLE III
APPLICATION OF `TagsCOMPkvk`

Key (k)	*Value (v)*	*Conflict (k)*
building	yes	**amenity**
building	yes	**leisure**
building	yes	**email**
building	yes	building:levels
building	yes	name
building	yes	wikidata
highway	traffic_signals	**bus**
highway	service	**layer**
highway	residential	**maxheight**
highway	residential	access
highway	crossing	bicycle
highway	service	covered
amenity	restaurant	**phone**
amenity	parking	**building**
amenity	pharmacy	**email**
amenity	pharmacy	wheelchair
amenity	restaurant	delivery
amenity	restaurant	takeaway
tourism	hotel	**phone**
tourism	hotel	**email**
tourism	hotel	**building**
tourism	hotel	website
tourism	hotel	wheelchair
tourism	artwork	start_date

Top-3 detected combinations. Errors in bold (the rest are false positives).

pattern as other ones included in the training set. We performed 19 manual overrides related to keys and values following the pattern *string:complement*. We are going to explain only two of them, but the other ones follow a similar structure.

The first manual override takes into account that the simultaneous occurrence of the (*building, yes*) tag and the *name* key is a typical false positive in `TagsCOMPkvk` (this combination is reported in Table III). In this case, *name* refers to the official name of the building. However, other tags (which are also false positives) refer to the official name of the building but in unofficial languages of the city. For example, this is the case with the simultaneous occurrence of the (*building*, *yes*) tag and keys such as *name:es* (in Spanish), *name:it* (in Italian), or *name:ch* (in Chinese). Therefore, as not all the official languages in the world have been used in the training set, we have considered that these tags will be automatically considered a false positive regardless of the suffix of the *name* key. This will allow our model to appropriately deal with different contexts, even if the language in question was not included in the training set. The second manual override, in this case labeling nodes as real errors, considers the simultaneous occurrence of the (*building*, *yes*) tag and the *currency:GBP* key. This is an error in `TagsCOMPkvk` and we know that the same pattern, using a different currency, will be also an error. For example, it is possible to find nodes having a (*building*, *yes*) tag and other tags having keys such as *currency:EUR* or *currency:USD*. Following the same reasoning as in the previous case, since not all the official currencies in the world have been used in the training set, we have considered that these tags will be automatically considered an error regardless of the suffix of the *currency* key.

VI. NEW MT FRAMEWORK

An MT framework basically proceeds as follows. An expert has to define MRs and they must be implemented. Then, the tester selects the MR that she would like to check and a set of source inputs. The inputs are applied to the system, the corresponding outputs are observed and the MR is invoked to check whether they are violated. If such a violation is observed then we report it. Otherwise, we iterate the process with the application of more inputs. Note that the generation of follow-up inputs must take into account the input of the previous iteration and the specific MR being used.

A graphical view of our MT framework can be seen in Fig. 2. Note that the *Design MR*, *Implementation* and *Runtime* phases are almost the same as in a classical MT framework without false positives. The only difference appears in the *Runtime* phase because we do not stop when we find an error. Instead, we collect potential errors and classify them. In a previous phase, the *design models* phase, the ML models described in the previous section were created: the tester applied MRs to maps, and the expert classified the potential errors between real and false positives and trained the models. This phase is completely transparent for the users of our framework.

We think that this schema can be used in other MT frameworks producing false positives if we simply replace the selection of maps by the selection of SUTs in the corresponding domain. We will discuss this idea further when we describe lines of future work.

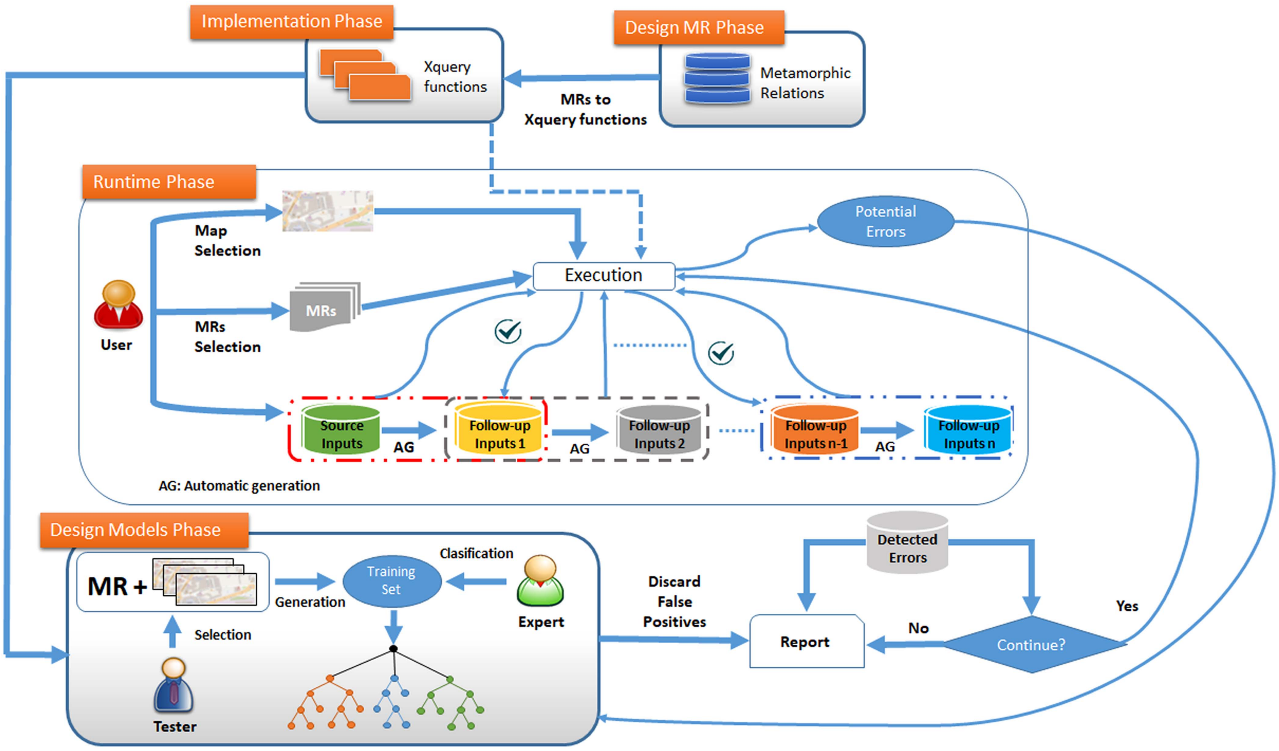


Fig. 2. Architecture of our MT framework.

Next, we explain a typical use of our framework by an external user. First, the user has to select the OSM map,[12] search the area to be checked and click the *Export* button. The OSM map is downloaded in XML format. Second, the user applies an MR to the map and obtains a list of potential errors (the XQuery implementation of our MRs can be found online.[13] Third, the user applies the ML model corresponding to the chosen MR and obtains a list of errors. The complete code, including training sets, can be found online.[14]

VII. Experiments and Evaluation

In this section, we will describe the results of our experiments. In particular, we will explain the main characteristics of the *training set*, that is, the data that we used to train our models, and of the *testing set*, the data that we used to assess the strength of our framework. First, we present the different research questions that we had before we started the evaluation of our framework and later on we will provide the answers to these questions in view of the results of the experiments. Finally, we will enumerate the detected threats to the validity of our results.

A. Research Questions

Our first research question is obvious.

Research Question 1: Does the combination of MT and ML provide a satisfactory solution?

If the answer to this question is positive, then we would like to further explore how *good* our proposal is. We would like to assess whether it is worth the effort of adding ML to our previous work where MT was the only used technique.

Research Question 2: Is it better to use the combination of MT and ML presented in this article or our previous work [12], which only uses MT?

In addition, we aim to determine whether the combination of MT and ML is superior to simply providing our ML models with all of the map information. It should be noted that in the former case, our MRs select only potentially problematic nodes, which carries the risk of overlooking some nodes.

Research Question 3: Is it better to use our approach or to apply our ML models to the whole map?

Our final research questions concern the generalizability of our framework to other cities. First, we would like to evaluate the usefulness of our approach when analyzing cities that were not included in the original training set.

Research Question 4: Does our approach work well with cities not included in the training set?

Second, we would like to evaluate whether having a generic model per MR, with many cities in the training set, provides better results than having several models per MR and country.

Research Question 5: Is it better to develop specific models for each country?

B. Training and Testing Sets

In this section we will describe in detail the training and testing sets.

[12] [Online]. Available: https://www.openstreetmap.org/
[13] [Online]. Available: https://github.com/jalmenUAL/MT-OSM
[14] [Online]. Available: https://github.com/MMH1997/MT-ML-OSM

TABLE IV
APPLICATION OF `TagsMISSkvk`

City	*building*	*highway*	*amenity*	*tourism*	*OSM Expert*
Almeria	80	69	38	12	199
Banja Luka	136	98	46	12	292
London	329	349	128	37	843
Madrid	24	280	116	56	692
Paris	199	262	123	32	616
Valencia	155	196	63	25	439
City of Mexico	260	235	78	34	607
New York	411	268	157	38	874
Sao Paolo	356	210	155	37	758
Nairobi	267	76	89	32	464
Tokyo	189	124	79	27	419
Total	**2622**	**2167**	**1072**	**342**	**6203**

Tags require labeling by an OSM expert on a city-by-city basis.

TABLE V
APPLICATION OF `TagsMISSkvk`

Key (k)	*Value (v)*	*Missing (k)*
building	house	**house**
building	yes	addr:city
building	yes	addr:housenumber
building	yes	addr:street
highway	crossing	**crossing**
highway	traffic_signals	**traffic_signals**
highway	footway	**footway**
highway	service	access
highway	bicycle	residential
highway	tertiary	surface
amenity	parking	**parking**
amenity	bench	**backrest**
amenity	waste_disposal	**waste**
amenity	restaurant	addr:city
amenity	restaurant	addr:housenumber
amenity	cafe	addr:postcode
tourism	artwork	**artwork_type**
tourism	information	**board_type**
tourism	hotel	addr:city
tourism	hotel	internet_access
tourism	artwork	artis_name

Tags require labeling by an OSM expert on a city-by-city basis.

TABLE VI
APPLICATION OF `TagsCOMPkv`

City	*building*	*highway*	*amenity*	*tourism*	*OSM Expert*
Almeria	156	375	145	7	683
Banja Luka	343	425	87	19	874
London	676	787	356	46	1865
Madrid	308	582	363	54	1307
Paris	322	809	356	35	1522
Valencia	470	489	206	23	1188
City of Mexico	359	678	289	49	1375
New York	618	579	378	34	1609
Sao Paolo	621	780	456	52	1909
Nairobi	629	590	198	33	1450
Tokyo	312	886	456	43	1697
Total	**4814**	**6980**	**3290**	**395**	**15 479**

Tags require labeling by an OSM expert on a city-by-city basis.

TABLE VII
APPLICATION OF `TagsCOMPkv`

Key (k)	*Value (v)*	*Conflict (k)*
building	**hospital**	amenity
building	**hotel**	tourism
building	**school**	amenity
building	residential	highway
building	residential	landuse
building	apartments	use
highway	**proposed**	building
highway	**commercial**	building
highway	traffic_signals	crossing
highway	crossing	traffic_signals
highway	footway	crossing
amenity	**pharmacy**	healthcare
amenity	**dentist**	healthcare
amenity	**clinic**	healthcare
amenity	parking	parking
amenity	pub	disused:amenity
amenity	recycling	waste
tourism	**park**	leisure
tourism	**disused**	historic
tourism	hotel	building
tourism	artwork	historic:tourism

Top-3 detected combinations. Errors in bold (the rest are false positives).

The training set includes eleven cities. In the selection process of the cities we took into account their size (which, usually, is directly proportional to the occurrences of the relevant keys and values combinations) and the location of the city, selecting at least one city in each continent (except in Oceania). The cities are listed in Table I and we have included the total number of tags matching keys of interest. The most frequent keys are *building* (62.1%), followed by *highway* (28.1%), *amenity* (8.9%), and *tourism* (0.9%). As we use a different model for each MR, we require three different training sets. These training sets are obtained by applying the MRs to the original map. After applying our MRs, we obtained a *reduced* version of the original map, which included only the potential errors related to the MR in question. Tables II (`TagsCOMPkv`), IV (`TagsMISSkvk`), and VI (`TagsCOMPkv`) show the number of occurrences corresponding to each key in the final training sets. The last column of each table shows the total number of occurrences per city: these are the nodes that the expert had to (manually) classify as either real or false positives. These tables also show the advantage of our combined MT and ML approach. Applying only MT, the expert needs to label all the tags of the three tables (around 30 000 tags). This tedious labeling process is what our framework is able to save. In order to highlight where the potential problems are, Tables III (`TagsCOMPkvk`), V (`TagsMISSkvk`), and VII (`TagsCOMPkv`) show the three combinations of source and follow-up inputs that produced the immense majority of errors (indicated in boldface) and false positives appearing in the different training sets.

The testing process included ten cities. In their selection we took into account the size and the location of the city, choosing at least one city per continent (again, except in Oceania). Furthermore, we chose cities that belonged to the same countries as certain cities in the training set, as well as cities from countries that were not represented in the training set. Table VIII shows the cities and the number of total tags of the original map matching the chosen inputs.

C. Metrics

In this section, we will define the metrics used to evaluate our model. We will use the following standard notation.

1) TP and TN denote, respectively, the number of true positives and negatives identified by the model. The elements in these sets were correctly classified by the model.

TABLE VIII
TESTED CITIES

City (Country)	*building*	*highway*	*amenity*	*tourism*
Europe				
Barcelona (Spain)	21 016	12 004	3648	426
Brussels (Belgium)	19 588	11 956	4279	380
Rome (Italy)	22 274	11 402	3985	546
America				
Boston (USA)	18 148	11 235	3421	339
Havana (Cuba)	20 482	9360	3962	520
Ottawa (Canada)	17 053	10 246	3664	117
Santiago (Chile)	18 563	9206	3957	362
Africa				
Cairo (Egypt)	16 632	10 376	2934	252
Luanda (Angola)	14 901	10 289	2258	143
Asia				
Wuhan (China)	16 878	11 710	3288	240

Distribution of analyzed tags.

2) FP and FN denote, respectively, the number of false positives and negatives identified by the model. Therefore, the model was not able to successfully classify the elements belonging to these sets.
3) Tot is equal to the total number of elements in the sample. Therefore, Tot=TP+TN+FP+FN.

The first metric, the *accuracy* of the model, measures the success percentage of the model. Formally, we define the *accuracy* of the model as

$$\frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{Tot}}$$

Our second metric, the *F1-Score*, combines two interesting characteristics of models: *recall*, that is, the ratio between the positives correctly classified and the total number of positives, and *precision*, that is, the ratio between the true positives and all the positives. Therefore, if we consider that the recall (R) and precision (P) of a model are given by the expressions

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

then, the *F1-score* is defined as

$$2 \cdot \frac{R \cdot P}{R + P}.$$

In the previous formulation, we applied the F1-Score to a binary-class classification model, where each class represented the possible outputs of the MRs. This is, in fact, the case of the models corresponding to the `TagsCOMPkvk` and `TagsCOMPkv` MRs because the outputs of these models are either true or false positive, that is, we have two classes. However, we require a slight modification to evaluate the model corresponding to the `TagsMISSkvk` MR because, in this case, we have a multiclass classification model. The `TagsMISSkvk` MR classifies key-value-key combinations in one of the following classes: required (in the sense of being mandatory), recommended and not-allowed. For example, consider a node having the $(amenity, parking)$ tag. This node must have another tag having the *parking* key (i.e., it is required), should have a tag having the *addr:street* key (i.e., it is recommended) and should not have a tag having *phone* as key (i.e., it is not-allowed). Something similar happens, for example, with tags such as $(amenity, restaurant)$ or (tourism, hotel).

TABLE IX
APPLICATION OF `TagsCOMPkvk`

City	*building*	*highway*	*amenity*	*tourism*	*Acc.*	*F1*
Barcelona	489	231	395	56	0.963	0.923
Brussels	415	324	453	43	0.952	0.927
Rome	312	267	391	61	0.968	0.952
Boston	345	216	281	32	0.929	0.912
Havana	443	314	453	63	0.942	0.907
Ottawa	265	267	387	16	0.971	0.956
Santiago	288	212	403	34	0.947	0.925
Cairo	301	276	344	31	0.947	0.925
Luanda	380	208	289	12	0.946	0.933
Wuhan	208	199	203	25	0.915	0.874
Total	**3446**	**2514**	**3599**	**373**	**0.948**	**0.923**

Generated combinations, accuracy and F1-score.

TABLE X
APPLICATION OF `TagsMISSkvk`

City	*building*	*highway*	*amenity*	*tourism*	*Acc.*	*F1**
Barcelona	368	267	126	34	0.952	0.952
Brussels	278	299	124	43	0.957	0.957
Rome	356	156	189	35	0.969	0.967
Boston	233	189	145	26	0.901	0.901
Havana	477	298	211	54	0.931	0.933
Ottawa	258	233	156	11	0.905	0.857
Santiago	305	208	188	29	0.927	0.924
Cairo	255	287	145	19	0.925	0.894
Luanda	289	269	114	14	0.950	0.943
Wuhan	104	156	99	12	0.917	0.933
Total	**2923**	**2362**	**1497**	**277**	**0.933**	**0.926**

Generated combinations, accuracy, and F1-Score.*

Formally, if we have n (with $n > 2$) different classes, then let F_i be the F1-score corresponding to the class i and N_i be the number of observations in the class i. We define the *F1-score** for a multiclass classification model as

$$\frac{1}{n} \sum_{i=1}^{n} \frac{F_i \cdot N_i}{Tot}.$$

As we have already said, in the case of `TagsMISSkvk` we have $n = 3$.

While there are other common metrics for classification tasks, such as precision or recall, we chose to use the *F1-score* as it considers both precision and recall in its calculation. In addition, the *F1-score* is more robust and less sensitive to unbalanced classes compared with other metrics and it is easier to interpret than precision and recall.

D. Experiments

In this section, we will show the results corresponding to the three MRs. We also discuss additional experiments that were performed in order to answer some of our research questions.

First, after applying the corresponding MRs, we obtain the three final testing sets. The number of tags by key after the application of our MRs is shown in the first four columns of Tables IX (`TagsCOMPkvk`), X (`TagsMISSkvk`), and XI (`TagsCOMPkv`). Note that these testing sets only include the potential errors obtained after the application of the MRs, that is, a reduced version of the original map (see Table VIII). The goal, as we explained in the introduction of the article, is to increase accuracy by targeting potentially problematic nodes, that is, a very small subset of the analyzed nodes. One of our experiments

TABLE XI
APPLICATION OF `TagsCOMPkv`

City	*building*	*highway*	*amenity*	*tourism*	*Acc.*	*F1*
Barcelona	429	705	342	45	0.989	0.975
Brussels	560	790	554	34	0.993	0.943
Rome	543	902	387	55	0.949	0.933
Boston	654	894	367	48	0.995	0.967
Havana	665	879	480	56	0.960	0.800
Ottawa	454	654	356	13	1	1
Santiago	546	812	432	44	0.967	0.912
Cairo	450	767	280	22	1	1
Luanda	405	899	212	19	1	1
Wuhan	290	678	254	22	0.902	0.782
Total	**4996**	**7980**	**3664**	**358**	**0.976**	**0.931**

Generated combinations, accuracy, and F1-Score.

TABLE XII
APPLICATION OF `TagsCOMPkvk`: NONMT VERSUS MT+ML

Training Process (NonMT)						
City	*building*	*highway*	*amenity*	*tourism*	*Total*	
Madrid	1190	1640	1220	85	4193	
Training Process (MT+ML)						
City	*building*	*highway*	*amenity*	*tourism*	*Total*	
Madrid	38	22	21	10	91	
Testing Process (NonMT)						
City	*building*	*highway*	*amenity*	*tourism*	*Acc.*	*F1*
Almeria	985	1464	932	122	0.836	0.792
Testing Process (MT+ML)						
City	*building*	*highway*	*amenity*	*tourism*	*Acc.*	*F1*
Almeria	26	25	39	29	0.925	0.895

compares the efficacy of our approach with respect to work with the whole map.

Table IX (last two columns) shows the results obtained for `TagsCOMPkvk`. Accuracies range from 0.915 up to 0.971, with an average of 0.948. Analyzing by continent, the results are better for cities located in Europe (an average of 0.961), followed by Africa and America with an average of 0.947. Asia, with an average of 0.915, is the continent where accuracy is lowest. We think that the combination of two factors explains slightly lower accuracies in Asia: only one Asian city was used in the training set and contributors used different alphabets in Asian languages. In addition, F1-scores range from 0.874 up to 0.956, with an average of 0.923. It follows a similar pattern to the accuracies by continent.

The results corresponding to `TagsMISSkvk` can be found in Table X (last two columns). In this case, accuracies range from 0.901 to 0.969, with an average of 0.933. By continent, the best results are again obtained in Europe (an average of 0.959), followed by Africa (0.938), and both America and Asia with an average of 0.916. This is the more stable model, concerning accuracy, across continents. In addition, F1-scores* range from 0.857 up to 0.967, with an average of 0.926. It follows by continent a similar pattern to the accuracies. F1-score* values are consistently similar to the accuracy values in almost all cases.

The results of `TagsCOMPkv` are given in Table XI (last two columns). In this case, accuracies range from 0.902 to 1.0 and we obtain a very high average: 0.98. By continent, the best results are obtained in Africa (an average of 1.0), followed by America (0.981), and Europe (0.977). Finally, the worst results appear in Asia with an average of 0.902. We believe that the reason for the lower accuracy, which is still above 0.90 and represents a very high value, is the same as the one mentioned in the case of `TagsCOMPkvk`. In addition, F1-scores range from 0.782 to 1.0. The pattern of the results by continent is approximately the same as for accuracies. However, in some cities, the F1-scores are significantly smaller than their respective accuracies, whereas in the cases that accuracies reach 1.0, the F1-scores also reach 1.0.

This first group of experiments, associated with the testing sets, show that our models obtain accuracies higher than 0.9. Therefore, we have chosen to conduct additional experiments to address our research questions. Our aim is to further scrutinize our framework and identify any potential weaknesses it may have.

We decided to analyze how useful was to combine ML and our original MT framework [12] with respect to simply using an ML approach. The rationale is that if the results are similar, then we can simply discard MT and use ML to analyze OSM. Next, we report on two experiments where we analyze potential errors related to the `TagsCOMPkvk` MR, considering the map of the city of Madrid to train the models and the city of Almería to test the resulting models. We denote by NonMT the first approach where we simply use ML, whereas the approach presented in this article, where MT is applied before ML models classify potential errors, is denoted by MT+ML. Table XII shows the number of tags of the analyzed maps matching the selected inputs (NonMT approach) and the number of tags matching the same inputs, in the same maps, but after applying the `TagsCOMPkvk` MR (MT+ML approach).

E. Answers to Research Questions

We can positively answer Research Question 1: Our approach is useful because it accurately separates false and true positives without human intervention. We can claim this because the initial set of experiments involving the testing sets demonstrated excellent performance of our trained RF model. Accuracy and F1-scores consistently exceeded 0.9 and reached a perfect score of 1.0 for some MRs and cities within the testing set. The specific values can be seen in the last two columns of Tables IX, X, and XI, where we can see, in particular, that the average for accuracy ranges between 0.933 and 0.976 and the average for F1-scores ranges between 0.923 and 0.931.

The positive answer to Research Question 1 brings us to Research Question 2. The answer to this question depends on the priorities and available resources. If we have a critical, and relatively small, map where we cannot allow any error of the types targeted by our MRs, then we should apply our MRs and an expert should manually discard false positives. In this very specific situation, the answer to Research Question 2 is negative. However, in general, the users of our framework will deal with large maps and either they will not have access to an expert or it will be too costly to have an expert checking a huge amount of potential errors. In this general case, the answer to these Research Questions is clear: Our approach is much better than the previous one because it does not need the intervention of an expert and can accurately separate true and false positives belonging to a large set.

The results of the experiments presented in Table XII provide a clear positive answer to Research Question 3: A combination of MT and ML is more effective than a single ML approach. The first observation, as expected by our experience analyzing OSM maps using MT, is that the number of tags considered in the MT+ML approach is much lower, staying below 5% of the total number, with the exception of tags including the *tourism* key because they are less frequent and we can expect a higher variance. Therefore, the MR acts as a filter in the learning process, allowing to more precisely define the training sets and, interestingly enough, obtaining better accuracy as we will see below. Specifically, in order to label the training set in the NonMT approach, the OSM expert needs to check 4193 tags, versus a total of 91 if the expert focuses on the potential problems detected by the application of the `TagsCOMPkvk` MR. We could expect that a larger training set would allow us to obtain better accuracy and F1-score when testing other cities, but this is not the case. Actually, in the first model we obtained a mere 0.836 accuracy and 0.792 F1-score versus a 0.925 accuracy and 0.895 F1-score obtained in the second model. Note that the first value is worse than any of the accuracies reported before; the second value is in line with those accuracies. The conclusion of these experiments is that MRs allow us to filter most of the suspicious cases (i.e., potential errors), to carry out the training process using only this reduced number of elements and allowing us to obtain a better accuracy in the testing process. Therefore, the approach presented in this article achieves a higher accuracy with a severe reduction of the number of (manually) checked tags. We also have a clear positive answer to Research Question 4: the obtained models work in cities that are not included in the training set. This was one of our main worries before we started the experiments: Our framework might work well in cities *similar* to the ones belonging to the training set but diminish its effectiveness in other cities. Fortunately, the performance was very good for cities in countries that were not included in the training set and even having very different languages. For example, consider the results obtained for Luanda (Angola) where we achieved excellent results (with accuracies ranging from 0.95 to 1.0 and F1-scores ranging from 0.933 to 1.0) despite having only one African city, Nairobi, in the training set, which is located in a different country (Kenya) and separated by more than 4 000 km, and only one Portuguese speaking city, Sao Paulo, located in a different continent.

Concerning Research Question 5, having a model per MR versus having specific models per MR and country, the hypothesis was that dedicated models might be able to target specific errors of the users of a country. In other words, we initially thought that a *specialized* model, even including a smaller number of cities in its training set, might be more accurate than a *general* model. In order to get a proper answer to our research question, we performed additional, noninitially planned, experiments that evaluated the impact of having dedicated models per country, that is, models that use as training set cities from the same country, instead of training sets corresponding to cities from different countries. We present in detail the results of one of the experiments (the other experiments gave similar results). We conducted an analysis on an area of Barcelona (a city that had already been used in our testing set) using a model that was trained only on Spanish cities included in the original training set (that is, Almeria, Valencia, and Madrid). The obtained accuracies (0.91 for `TagsCOMPkvk` and `TagsCOMPkv` MRs, and 0.84 for the `TagsMISSkvk` MR) are lower than the ones obtained when using the whole training set (0.963 for `TagsCOMPkvk`, 0.952 for `TagsMISSkvk`, and 0.989 for `TagsCOMPkv`). Therefore, the inclusion of new cities in the training set did not lead to a deterioration of the models, avoiding the risk of overfitting. This happens because new training data (from new cities) will always give new information to the model without affecting the previous information. In conclusion, we found out that errors are more similar across cities and countries than we thought and, therefore, it is better to use models trained with as many cities as possible, providing a negative answer to Research Question 5.

F. Threats to the Validity of the Results

Next, we analyze the potential threats to the validity of the results presented in the previous section. As usual, we have three categories.

1) Threats to *internal validity:* These are threats associated with uncontrolled factors that might have an influence on the results.
2) Threats to *external validity:* In this group we study whether we can used the lessons learned from our experiments in other environments.
3) Threats to *construct validity:* Finally, we analyze whether our framework can be useful.

The main concern when analyzing threats to internal validity is to discard the presence of faults in the code. In our case, we have two different sources of potential problems: the implementation of the MT framework and the creation of the ML models associated with each MR. Concerning the first case, we build on top of our previous work, both on the application of MT to OSM and on OSM itself. Therefore, we simply reused the code to prepare and process maps and to apply MRs to maps. The code was thoroughly tested and was used in several experiments before we used it to carry out the experiments of this article. We would like to mention that as part of the testing process, in our previous work [12] we used MT to test the novel feature of our MRs: the use of thresholds. We also paid attention to validate the uninterrupted generation of follow-up inputs because this is the main feature to ensure most of the automation during the application of our MRs. Finally, in order to increase its effectiveness, we applied a hyperparameter optimization in the RF models. The second concern is associated with the choice of our MRs and our ML model. First, if we do not use MRs with strong capabilities to find errors, then the usefulness of our framework will be strongly reduced. Second, if we choose an ML model that it is not suitable to accurately discard false positives, then we will not be achieving our main goal. Our MRs were carefully designed in our previous work and discussed with colleagues working either on MT or on OSM. We chose RF as our model after our initial experiments showed that they were better suited for our goal than simple classification trees. Other alternatives were discussed with colleagues working on ML and discarded in favor of the selected model.

In the group including threats to external validity, our first priority was to analyze whether our framework was able to properly work in different situations. As we have already commented, when choosing maps for training and testing we were increasing, as much as possible, the heterogeneity of the considered cities. In particular, we chose cities in different countries and continents so that we reduce the impact of bad OSM contributors using very different languages. The second threat is related to the generalizability of our framework to analyze other types of maps. We have inherited this threat from our previous work and, therefore, this threat was already studied. We believe that we are able to deal with any mapping system providing access to its data. Unfortunately, commercial applications such as Bing, Google, Wikimapia, and Yandex do not allow us to access data and, therefore, we cannot appropriately address this threat.

Finally, in the group including threats to construct validity we first considered three aspects: the error detection ability of the seeds (source inputs), the performance of the MRs in different areas, and the reliability of the models to identify real errors versus false positives. In order to mitigate this threat, we used areas in different cities/countries and intensively *evaluated* the different MRs, and their associated ML models, by executing a wide range of experiments that were able to cover the analyzed areas. Since we use XQuery, the compliant XQuery 3.1 processor BaseX and the Scikit-Learn 1.1.2 Python library, there is a clear dependence of our framework on external programs. However, XQuery is a standardized query language[15] recommended by the World Wide Web Consortium (W3C), with many practitioners and wide acceptance in the industry, whereas Scikit-Learn is widely used to implement ML models in Python.

It is worth noting that the code and maps used in this article are freely available online.[16] We hope that this helps reducing the potential threats since any user of our framework can independently check, replicate, and analyze our findings.

VIII. Conclusion

We have presented a novel MT framework where we use ML to *remove* the OSM expert that was needed to decide whether a potential error in the map was a true or a false positive. We can claim that ML is a very useful tool in detecting false positives when MT is applied to find tagging mistakes in OSM. However, some questions may arise about the proposed method and we would like to discuss them here.

A plausible question is whether other domains—nonrelated to either maps or tags—could use ML in combination with MT. We think that as long as MRs can produce false positives, ML can help to detect them if they follow similar patterns. Domains in which thresholds are used in MRs are the more promising ones because false positives tend to show up and this is one of our lines of future work. We have performed some preliminary experiments, with good results, in a previous work where thresholds were encoded into MRs to evaluate cloud configurations [16]. We also think that the application of MT to debuggers [27] is a very good candidate because the percentage of false positives is high and certain patterns may detect them.

The second question is why should we use MT and ML instead of just using ML in isolation. Let us assume that ML is a good candidate for detecting errors, defects, and mistakes in a certain system. In this case, it should be possible to build an ML model for detecting these issues. However, our experience indicates that one should try and use MT as a tool for improving the process because MT can filter most of the errors and ML can be used for the most complex ones. One can argue that ML might be enough for all the errors. However, this is not true because ML can produce false positives too. Also note that the precision of ML depends, in general, on the training set. As an illustrative example, let us suppose that we train an ML model to detect both cars and zebras in images and the training set contains both images of cars and zebras. It is very likely that the ML model confuses cars with stripes and zebras. Thus, a filtering of the training set removing zebras might be more precise in the detection of cars. In other words, ML will work better if we just look for false positives instead of looking for errors in general. In addition, it should be noted that without the use of MT, the manual labeling task of the training set often uncovers common errors that follow repeated patterns. These errors are likely to not contribute to the improvement of the ML model. As a matter of fact, in this article we reported on an experiment showing that the combination of MT and ML produced better results than the use of ML. There is another work that utilizes MT to enhance ML [45], [60]. The authors improve the performance of a supervised Artcode classifier based on conventional classification methods by applying MRs, resulting in an MR-enhanced classifier. To achieve this, the MRs are used to generate new inputs corresponding to a set of blocks obtained by splitting the original image. In contrast, our proposal uses MRs to filter the training sets of the models by considering only potential problems related to each MR.

Another question is about the domain in which MT has been applied, how specific the domain is—maps and tags—and whether there are other similar domains in which the same kind of approach can be followed. This is our second main line of future work. At this point, we think that other systems in which tagging is crucial for a correct performance can be subject of a similar study. Currently, tags have become central in many systems, in the form of hashtags (Twitter, Instagram, TikTok, and Flickr) or key-value pairs playing the same role as in OSM (Wikipedia, DBpedia, and Wikidata). These systems suffer the same problems as OSM because anonymous users contribute to the tagging of objects (pictures, videos, and real world entities). For a correct performance of searching tools in such systems, a correct tagging is required.

A final question is how other users can use our framework to either analyze a different map or extend it with new MRs. As commented before, our trained models can be used in any map of the world, requiring only that the chosen MRs have been applied so that the user can send the suspicious nodes to

[15] [Online]. Available: https://www.w3.org/XML/Query/

[16] [Online]. Available: https://github.com/jalmenUAL/MT-OSM and https://github.com/MMH1997/MT-ML-OSM

the corresponding models. Second, a *metamorphic tester* might need to design a new MR to analyze OSM from a different perspective. The first step is to formally define the MR and implement it by following the pattern of our implementations, which can be found online.[17] The application of the MR to a map will return some suspicious elements (including false positives). The next step is to create a new ML model following the pattern given online.[18] First, it is necessary to tag suspicious elements by marking them as either real errors or false positives. This set will be the training set of the new ML model. After training, the resulting ML model can serve to detect real errors in the whole output of the MR when applied to a map. Let us emphasize that in order to obtain reliable results, the training set should be as large as possible and include the results of the application of the new MR to different maps.

Acknowledgment

The authors would like to thank the anonymous reviewers for the careful reading, interesting comments, and suggestions that have helped us to further strengthen the article.

References

[1] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed. Hoboken, NJ, USA: Wiley, 2011.

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

[3] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Trans. Softw. Eng.*, vol. 41, no. 5, pp. 507–525, May 2015.

[4] T. Y. Chen, S. C. Cheung, and S. M. Yiu, "Metamorphic testing: A new approach for generating next test cases," Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, Tech. Rep. HKUST-CS98-01, 1998.

[5] 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.

[6] 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.

[7] S. Elwood, "Volunteered geographic information: Key questions, concepts and methods to guide emerging research and practice," *GeoJournal*, vol. 72, no. 3–4, pp. 133–135, 2008.

[8] J. M. A.-Jiménez and A. B.-Terón, "Analyzing the tagging quality of the spanish OpenStreetMap," *ISPRS Int. J. Geo- Inf.*, vol. 7, no. 8, 2018, Art. no. 323.

[9] J. M. A.-Jiménez, A. B.-Terón, and M. Torres, "Integrating and querying OpenStreetMap and linked geo open data," *Comput. J.*, vol. 62, no. 3, pp. 321–345, 2019.

[10] J. M. A.-Jiménez, A. B.-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, p. 22, 2021.

[11] J. M. A.-Jiménez, A. B.-Terón, M. G. Merayo, and M. Núñez, "Metamorphic testing of OpenStreetMap," *Inf. Softw. Technol.*, vol. 138, 2021, Art. no. 106631.

[12] J. M. A.-Jiménez, A. B.-Terón, M. G. Merayo, and M. Núñez, "Using metamorphic testing to improve the quality of tags in OpenStreetMap," *IEEE Trans. Softw. Eng.*, vol. 49, no. 2, pp. 549–563, Feb. 2023.

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

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

[15] 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.

[16] 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.

[17] N. Mouha, M. S. Raunak, D. R. Kuhn, and R. Kacker, "Finding bugs in cryptographic hash function implementations," *IEEE Trans. Rel.*, vol. 67, no. 3, pp. 870–884, Sep. 2018.

[18] Z. Zhang et al., "DeepBackground: Metamorphic testing for deep-learning-driven image recognition systems accompanied by background-relevance," *Inf. Softw. Technol.*, vol. 140, 2021, Art. no. 106701.

[19] 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.

[20] R. Abreu, J. P. Fernandes, L. Llana, and G. Tavares, "Metamorphic testing of oracle quantum programs," in *Proc. 3rd Int. Workshop Quantum Softw. Eng.*, 2022, pp. 16–23.

[21] 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.

[22] 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.

[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] R. Guderlei and J. Mayer, "Statistical metamorphic testing testing programs with random output by means of statistical hypothesis tests and metamorphic testing," in *Proc. 7th Int. Conf. Qual. Softw.*, 2007, pp. 404–409.

[25] C. Murphy, K. Shen, and G. Kaiser, "Automatic system testing of programs without test oracles," in *Proc. 8th Int. Symp. Softw. Testing Anal.*, 2009, pp. 189–200.

[26] S. Segura, J. Troya, A. Durán, and A. R.-Cortés, "Performance metamorphic testing: Motivation and challenges," in *Proc. IEEE/ACM 39th Int. Conf. Softw. Eng.: New Ideas, Emerg. Technol. Results Track*, 2017, pp. 7–10.

[27] S. Tolksdorf, D. Lehmann, and M. Pradel, "Interactive metamorphic testing of debuggers," in *Proc. 28th ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2019, pp. 273–283.

[28] 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.

[29] 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.

[30] 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.

[31] J. Brown, Z. Q. Zhou, and Y.-W. Chow, "Metamorphic testing of mapping software," in *Towards Integrated Web, Mobile, and IoT Technology*, Berlin, Germany: Springer, 2019, pp. 1–20.

[32] 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.

[33] J. E. V.-Munoz, S. Srivastava, D. Tuia, and A. X. Falcao, "OpenStreetMap: Challenges and opportunities in machine learning and remote sensing," *IEEE Geosci. Remote Sens. Mag.*, vol. 9, no. 1, pp. 184–199, Mar. 2021.

[34] Q. Li, H. Fan, X. Luan, B. Yang, and L. Liu, "Polygon-based approach for extracting multilane roads from OpenStreetMap urban road networks," *Int. J. Geographical Inf. Sci.*, vol. 28, no. 11, pp. 2200–2219, 2014.

[35] Y. Xu, Z. Xie, L. Wu, and Z. Chen, "Multilane roads extracted from the OpenStreetMap urban road network using random forests," *Trans. GIS*, vol. 23, no. 2, pp. 224–240, 2019.

[36] N. Karagiannakis, G. Giannopoulos, D. Skoutas, and S. Athanasiou, "OSMRec tool for automatic recommendation of categories on spatial entities in OpenStreetMap," in *Proc. 9th ACM Conf. Recommender Syst.*, 2015, pp. 337–338.

[37] A. L. Ali, F. Schmid, R. A.-Salman, and T. Kauppinen, "Ambiguity and plausibility: Managing classification quality in volunteered geographic information," in *Proc. 22nd ACM Int. Conf. Adv. Geographic Inf. Syst.*, 2014, pp. 143–152.

[17] [Online]. Available: https://github.com/jalmenUAL/MT-OSM

[18] [Online]. Available: https://github.com/MMH1997/MT-ML-OSM

[38] N. Tempelmeier and E. Demidova, “OVID: A machine learning approach for automated vandalism detection in OpenStreetMap,” in *Proc. 29th ACM Int. Conf. Adv. Geographic Inf. Syst.*, 2021, pp. 415–418.

[39] D. Feldmeyer, C. Meisch, H. Sauter, and J. Birkmann, “Using OpenStreetMap data and machine learning to generate socio-economic indicators,” *ISPRS Int. J. Geo-Inf.*, vol. 9, no. 9, 2020, Art. no. 498.

[40] K. T. Jacobs and S. W. Mitchell, “OpenStreetMap quality assessment using unsupervised machine learning methods,” *Trans. GIS*, vol. 24, no. 5, pp. 1280–1298, 2020.

[41] S. Keller, R. Gabriel, and J. Guth, “Machine learning framework for the estimation of average speed in rural road networks with OpenStreetMap data,” *ISPRS Int. J. Geo- Inf.*, vol. 9, no. 11, 2020, Art. no. 638.

[42] X. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, “Testing and validating machine learning classifiers by metamorphic testing,” *J. Syst. Softw.*, vol. 84, no. 4, pp. 544–558, 2011.

[43] A. Dwarakanath et al., “Identifying implementation bugs in machine learning based image classifiers using metamorphic testing,” in *Proc. 27th ACM SIGSOFT Int. Symp. Softw. Testing Anal.*, 2018, pp. 118–128.

[44] M. Jia, X. Wang, Y. Xu, Z. Cui, and R. Xie, “Testing machine learning classifiers based on compositional metamorphic relations,” *Int. J. Performability Eng.*, vol. 16, no. 1, pp. 67–77, 2020.

[45] L. Xu, D. Towey, A. P. French, S. Benford, Z. Q. Zhou, and T. Y. Chen, “Using metamorphic relations to verify and enhance artcode classification,” *J. Syst. Softw.*, vol. 182, 2021, Art. no. 111060.

[46] A. Dwarakanath, M. Ahuja, S. Podder, S. Vinu, A. Naskar, and M. V. Koushik, “Metamorphic testing of a deep learning based forecaster,” in *Proc. 4th Int. Workshop Metamorphic Testing*, 2019, pp. 40–47.

[47] X. Xie, Z. Zhang, T. Y. Chen, Y. Liu, P.-L. Poon, and B. Xu, “METTLE: A METamorphic testing approach to assessing and validating unsupervised machine LEarning systems,” *IEEE Trans. Rel.*, vol. 69, no. 4, pp. 1293–1322, Dec. 2020.

[48] J. D. Ellis, R. Iqbal, and K. Yoshimatsu, “Verification of the neural network training process for spectrum-based chemical substructure prediction using metamorphic testing,” *J. Comput. Sci.*, vol. 55, 2021, Art. no. 101456.

[49] Q.-H. Luu, M. F. Lau, S. P. H. Ng, and T. Y. Chen, “Testing multiple linear regression systems with metamorphic testing,” *J. Syst. Softw.*, vol. 182, 2021, Art. no. 111062.

[50] U. Kanewala and J. M. Bieman, “Using machine learning techniques to detect metamorphic relations for programs without test oracles,” in *Proc. 24th Int. Symp. Softw. Rel. Eng.*, 2013, pp. 1–10.

[51] U. Kanewala, J. M. Bieman, and A. B.-Hur, “Predicting metamorphic relations for testing scientific software: A machine learning approach using graph kernels,” *Soft. Testing, Verification, Rel.*, vol. 26, no. 3, pp. 245–269, 2016.

[52] B. Hardin and U. Kanewala, “Using semi-supervised learning for predicting metamorphic relations,” in *Proc. 3rd Int. Workshop Metamorphic Testing*, 2018, pp. 14–17.

[53] 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.

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

[55] 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/

[56] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, pp. 5–32, 2001.

[57] F. Farris, “The gini index and measures of inequality,” *Amer. Math. Monthly*, vol. 117, pp. 851–864, 2010.

[58] N. Prasad, P. Kumar, and N. Mm, “An approach to prediction of precipitation using gini index in SLIQ decision tree,” in *Proc. 4th Int. Conf. Intell. Syst., Modell., Simul.*, 2013, pp. 56–60.

[59] X. Zhang, C. Huang, and H. Zhang, “The application of gini coefficient in regional environmental pollutants distribution plan,” in *Proc. 2nd Conf. Environ. Sci., Inf. Appl. Technol.*, 2010, pp. 426–429.

[60] L. Xu, D. Towey, A. P. French, S. Benford, Z. Q. Zhou, and T. Y. Chen, “Enhancing supervised classifications with metamorphic relations,” in *Proc. 3rd Int. Workshop Metamorphic Testing*, 2018, pp. 46–53.

Manuel Méndez received two master’s degrees in mathematics and data science and big data from the University of Sevilla, in 2021, and the Ph.D. degree in computer science, under the supervision of M. G. Merayo.

His research interests include the application of machine learning techniques to different unrelated fields, such as forecasting (in particular, air quality and traffic) and software testing.

Antonio Becerra-Terón received the Ph.D. degree in computer science from the University of Almería, in 2003.

He is currently an Associate Professor with the Informatics Department, the University of Almeria, Almeria, Spain. He has authored or coauthored more than 30 papers in international journals and conferences. His research interests include query database languages, fuzzy programming, software testing, in particular, OpenStreetMap, XQuery, RDF, and SPARQL.

Jesús M. Almendros-Jiménez received the Ph.D. degree in mathematics from the Complutense University of Madrid, in 1999.

He is currently a Professor with the Universidad de Almería (UAL), Almería, Spain, where he is the leader of the Information Systems Research Group. His research interests include declarative programming, database query languages, fuzzy systems, and geographic information systems, mainly focused in semantics and implementation of programming languages and transformation, debugging and testing tools.

Mercedes G. Merayo Ph.D. degree in computer science from the Complutense University of Madrid, in 2009.

She is currently a Professor with the Universidad Complutense de Madrid, Madrid, Spain. She leads the Design and Testing of Reliable Systems research group.

Prof. Merayo is a member of the SVT - ACM/SIGAPP Symposium on Applied Computing Steering Committee and the Chair of the IFIP - ICTSS Committee Steering Committee. She has belonged to more than 100 programme committees of international events. She is a member of the editorial board of the *Journal of Information and Telecommunication*.

Manuel Núñez received the Ph.D. degree in mathematics from the Complutense University of Madrid, in 1996.

He is currently a Professor of computer science with the Complutense University of Madrid, Madrid, Spain.

Prof. Núñez is a Member of two IEEE Technical Committees, the Board of Directors of the Tarot Summer School on Software Testing, and the A-MOST, ICCCI, and QRS Steering Committees. He is a member of the editorial board of *Software Testing, Verification and Reliability, Transactions on Computational Collective Intelligence,* and *Vietnam Journal of Computer Science*.