Semantic Web 0 (2017) 1
IOS Press

Machine Learning in the Internet of Things: a Semantic-enhanced Approach

Michele Ruta [a,*], Floriano Scioscia [a], Giuseppe Loseto [a], Agnese Pinto [a], and Eugenio Di Sciascio [a]
[a] *Polytechnic University of Bari, Department of Electrical and Information Engineering, via E. Orabona 4, I-70125, Bari, Italy*
E-mail: name.surname@poliba.it

Abstract. Novel Internet of Things (IoT) applications and services rely on an intelligent understanding of the environment leveraging data gathered via heterogeneous sensors and micro-devices. Though increasingly effective, Machine Learning (ML) techniques generally do not go beyond classification of events with opaque labels, lacking machine-understandable representation and explanation of taxonomies. This paper proposes a framework for semantic-enhanced data mining on sensor streams, amenable to resource-constrained pervasive contexts. It merges an ontology-based characterization of data distributions with non-standard reasoning for a fine-grained event detection. The typical classification problem of ML is treated as a resource discovery by exploiting semantic matchmaking. Outputs of classification are endowed with computer-processable descriptions in standard Semantic Web languages, while explanation of matchmaking outcomes motivates confidence on results. A case study on road and traffic analysis has allowed to validate the proposal and achieve an assessment with respect to state-of-the-art ML algorithms.

Keywords: Semantic Web, Machine Learning, Non-standard Reasoning, Internet of Things

1. Introduction

The Internet of Things (IoT) paradigm is emerging through the widespread adoption of sensing micro- and nano-devices dipped in everyday environments and interconnected in low-power, lossy networks. The amount and consistency of pervasive devices increases daily and then the rate of raw data available for processing and analysis grows up exponentially. More than ever, effective methods are needed to treat data streams with the final goal of giving a meaningful interpretation of retrieved information.

The *Big Data* label has been coined to denote the research and development of data mining techniques and management infrastructures to deal with "volume, velocity, variety and veracity" issues [33] emerging when very large quantities of information materialize and need to be manipulated. Hence, Machine Learning (ML) is adopted to classify raw data and make predictions oriented to decision support and automation. Progress in ML algorithms and optimization goes hand in hand with advances in pervasive technologies and Web-scale data management architectures, so that undeniable benefits have been produced in data analysis. Nevertheless, some non-negligible weaknesses are still evident with respect to the increasing complexity and heterogeneity of pervasive computing scenarios. Particularly, the lack of machine-understandable characterization of outputs is a prominent limit of state-of-the-art ML techniques for a possible exploitation in fully autonomic application scenarios.

This paper proposes a framework named *MAFALDA*[1] (as *MAtchmaking Features for mAchine Learning Data Analysis*), aiming to enhance classical ML analysis on IoT data streams, by associating semantic descriptions to information retrieved from the physical world, as

*Corresponding author. E-mail: michele.ruta@poliba.it.

[1] The name should give a *retcon* with the well-known Quino comic strip to hint at the shrewd gaze of Mafalda character with her investigating attitude to life and her curiosity about the world.

1570-0844/17/$35.00 © 2017 – IOS Press and the authors. All rights reserved

opposed to simplistic classification labels. The basic idea is to treat a typical ML classification problem like knowledge-based resource discovery. This process calls for building a logic-based characterization of statistical data distributions and performing a fine-grained event detection through non-standard reasoning services for matchmaking [50].

The proposal leverages both general theory and technologies of Pervasive Knowledge-Based Systems (PKBS), intended as KBS whose individuals (assertional knowledge) are physically tied to objects disseminated in a given environment, without centralized coordination. Each annotation refers to an ontology providing the conceptualization and vocabulary for the particular knowledge domain and an advanced matchmaking can operate on the above metadata stored in sensing and capturing devices. No fixed knowledge bases are needed. In other words, inference tasks are distributed among devices which provide minimal computational capabilities. Stream reasoning techniques provide the groundwork to harness the flow of annotation updates inferred from low-level data, in order to enable proper context-aware capabilities. Along this vision, innovative analysis methods applied to data extracted by inexpensive off-the-shelf sensor devices can provide useful results in event recognition without requiring large computational resources: limits of capturing hardware could be counterbalanced by novel software-side data interpretation approaches.

MAFALDA has been tested and validated in a case study for road and traffic monitoring on a real data set collected for experiments. Results have been compared to classic ML algorithms in order to evaluate the provided performances. The experimental test campaign allows a preliminary assessment of both feasibility and sustainability of the proposed approach.

The remainder of the paper is as follows. Section 2 outlines motivation for the proposal, before discussing in Section 3 both background and state of the art on semantic data mining and ML for the IoT. The MAFALDA framework is presented in Section 4, while Section 5 and Section 6 report on the case study and the experiments, respectively. Conclusion finally closes the paper.

2. Motivation

Motivation for the work derives from the evidence of current limitations featuring the typical IoT scenarios. There, information is gathered through micro-devices attached to common items or deployed in given environments and interconnected wirelessly. Basically, due to their small size, such *objects* have minimal processing capabilities, small storage and low-throughput communication capabilities. They continuously produce raw data whose volume requires to be processed by advanced remote infrastructures. Classical ML techniques have been largely used for that, but often representations of detected events are not completely manageable in practical applications: this is mostly due to the difficulty of making descriptions interoperable with respect to shared vocabularies. In addition, usually ML solutions are very much tailored (*i.e.*, trained) to a specific classification problem. In spite of increasing device pervasiveness (miniaturization) and connectivity (interconnection capability), data streams produced at the edge of the network cannot be fully analyzed locally yet. Commonly adopted data mining techniques have two main drawbacks: i) they basically carry out no more than a classification task and ii) their precision increases if they are applied on very big data amounts, so on-line analyses hardly achieve high performance on typical IoT devices, due to computational and storage requirements. These factors still prevent the possibility of actualizing *thinking things*, able to make decisions and take actions locally after the sensing stage.

IoT relevance could be enhanced by annotating real-world objects, the data they gather and the environments they are dipped in, with concise, structured and semantically rich descriptions. The combination of the IoT with Semantic Web models and technologies is bringing about the so-called *Semantic Web of Things* (SWoT) vision, introduced in [49] and developed, *e.g.*, in [41,46,16,59]. This paradigm aims to enable novel classes of intelligent applications and services grounded on Knowledge Representation (KR), exploiting semantic-based automatic inferences to derive implicit information from an explicit event and context detection [48]. By associating a machine-understandable, structured description in standard Semantic Web languages, each classification output could have an unambiguous meaning. Furthermore, semantic-based explanation capabilities allow increasing confidence in system outcomes. If pervasive micro-devices are capable of efficient on-board processing on locally retrieved data, they can describe themselves and the context where they are located toward external devices and applications. This enhances interoperability and flexibility and enables autonomic-

ity of pervasive knowledge-based systems, not yet allowed by typical IoT infrastructures.

Two important consequences ensue. First of all, human-computer interaction could be improved, by reducing the user effort required to benefit from computing systems. In classical IoT paradigms, a user explicitly interacts with one device at a time to perform a task. On the contrary, user agents –running on mobile computing devices– should be able to interact simultaneously with many micro-components, providing users with context-aware personalized task and decision support. Secondly, even if ML techniques, algorithms and tools have enabled novel classes of analyses (particularly useful in the Big Data IoT perspective), the exploitation of logic-based approximate discovery strategies as proposed in the present work compensates possible faults in data capture, device volatility and unreliability of wireless communications. This supports novel, resilient and versatile IoT solutions.

3. Background

Hereafter main notions on Machine Learning and Description Logics are briefly recalled in order to make the paper self-contained and easily understandable. Then, most relevant related work is surveyed.

3.1. Basics of Machine Learning

Machine Learning (ML) [58] is a branch of Artificial Intelligence which aims to build systems capable of learning from past experience. ML algorithms and approaches are usually data-driven, inductive and general-purpose in nature; they are adopted to make predictions and/or decisions in some class of tasks, *e.g.*, spam filtering, handwriting recognition or activity detection. Basically, ML problems can be grouped in three categories: *classification*[2], *regression* and *clustering*. This paper focuses on classification, *i.e.*, the association of an observation (*sample*) to one of a set of possible categories (*classes*) –*e.g.*, whether an e-mail message is spam or not– based on values of its relevant attributes (*features*). Classification has therefore a discrete n-ary output.

The implementation of a ML system typically includes *training* and *testing* stages, respectively devoted to build a *model* of the particular problem inductively from *training data* and to system validation. Evaluation of classification performance is based on considering one of the output classes as the *positive* class and defining:

- *true positives (TP):* the number of samples correctly labeled as in the positive class;
- *false positives (FP):* the number of samples incorrectly labeled as in the positive class;
- *true negatives (TN):* the number of samples correctly labeled as not in the positive class;
- *false negatives (FN):* the number of samples incorrectly labeled as not in the positive class.

The following performance metrics for binary classification are often adopted:

- *Precision* (*a.k.a.* positive predictive value), defined as $P = \frac{TP}{TP+FP}$
- *Recall* (*a.k.a.* sensitivity), defined as $R = \frac{TP}{TP+FN}$
- *F-Score*, defined as the harmonic mean of precision and recall: $F = \frac{2PR}{P+R}$
- *Accuracy*, defined as $A = \frac{TP+TN}{TP+FP+TN+FN}$

Multiclass generalizations of the above formulas [52] have been also adopted in this paper.

Each available dataset to be classified is split in a *training set* for model building and a *test set* for validation. There exist several approaches for properly selecting training and test components. Among others, in *k-fold cross-validation*, the dataset is partitioned in k subsets of equal size; one of them is used for testing and the remaining $k-1$ for training. The process is repeated k times, each time using a different subset for testing. The simpler *holdout* method, instead, divides the dataset randomly, usually assigning a larger proportion of samples to the training set.

As detailed in the further Section 6, the performance of classification in the approach proposed here has been compared to the following popular ML techniques:

- *C4.5 decision tree* [43]: it adopts a greedy top-down approach for building a classification tree, starting from the root node. At each node, the information gain for each attribute is calculated and the attribute with the highest score is selected.
- *Functional Tree* [13,26]: a classification tree with logistic regression functions in the inner nodes and leaves. The algorithm can deal with binary and multiclass target variables, numeric and nominal attributes, and missing values.

[2] It should not be mistaken for the same-name problem in ontology management, consisting of finding all the implicit hierarchical relationships among concepts in a taxonomy.

- *Random Tree* [40]: it combines two ML algorithms, namely model trees and random forests, in order to achieve both robustness and scalability. Model trees are decision trees where every leaf holds a linear model optimized for the local subspace of that leaf. Random forests follow an *ensemble learning* approach which builds several decision trees and picks the mode of their outputs.
- *K-Nearest Neighbors, KNN* [2], is an instance-based learning algorithm. It locates the k instances nearest to the input one and determines its class by identifying the single most frequent class label. It is generally considered not tolerant to noise and missing values. Nevertheless, KNN is highly accurate, insensitive to outliers and it works well with both nominal and numerical features.
- *Multilayer Perceptron* [22]: a feedforward Artificial Neural Network (ANN), consisting of at least three layers of neurons with a nonlinear activation function: one for inputs, one for outputs and one or more *hidden* layers. Training is carried out through *backpropagation*. The *Deep Neural Network* (DNN) name characterizes ANNs having more than one hidden layer and using gradient descent methods for error reduction in backpropagation.

3.2. Basics of Description Logics

Description Logics –also known as Terminological languages, Concept languages– are a family of logic languages for Knowledge Representation in a decidable fragment of the First Order Logic [4]. Basic DL syntax elements are:

- *concepts* (*a.k.a. class*) names, standing for sets of objects, *e.g.*, *vehicle, road, acceleration*;
- *roles* (*a.k.a. object property*) names, linking pairs of objects in different concepts, like *hasTire, hasTraffic*;
- *individuals* (*a.k.a. instances*), special named elements belonging to concepts, *e.g.*, *Peugeot_207, Highway_A14*.

A semantic *interpretation* $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ consists of a *domain* Δ and an *interpretation function* $\cdot^{\mathcal{I}}$ which maps every concept to a subset of Δ, every role to a subset of $\Delta \times \Delta$, every individual to an element of Δ.

Syntax elements can be combined using *constructors* to build concept and role *expressions*. Each DL has a different set of constructors. Concept expressions can be used in *inclusion* and *definition* axioms, which model knowledge elicited for a given domain by restricting possible interpretations. A set of such axioms is called *Terminological Box (TBox)*, *a.k.a. ontology*. Semantics of inclusions and definitions is based on set containment: an interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies a definition $C \equiv D$ when $C^{\mathcal{I}} = D^{\mathcal{I}}$. A *model* of a TBox $\mathcal{T}$ is an interpretation satisfying all inclusions and definitions in $\mathcal{T}$. A set of axioms on individuals (*a.k.a.* facts) sets up an *Assertion Box (ABox)*, which composes a *Knowledge Base KB* together with its reference TBox.

Adding new constructors makes DL languages more expressive. Nevertheless, this usually leads to a growth in computational complexity of inference services [6]. This paper refers specifically to the *Attributive Language with unqualified Number restrictions* ($\mathcal{ALN}$) DL. It provides adequate expressiveness to support the modeling patterns described in Section 4.1, while granting polynomial complexity to both standard and non-standard inference services. Syntax and semantics of $\mathcal{ALN}$ constructors are reported in Table 1, along with the corresponding elements in the RDF/XML serialization of the Web Ontology Language (OWL 2)[3]. OWL also supports *annotation properties* associated to class and property names, *e.g.*, for comments and versioning information.

3.3. Related work

The semantic-enhanced machine learning approach proposed here is basically general-purpose, but it has been particularly devised for the Internet of Things. In IoT scenarios, smart interconnected objects gather environmental data samples, useful to identify and predict many real-world phenomena which exhibit patterns: early research has shown state-of-the-art ML is effective in the domain of ubiquitous sensor networks [35], although carried experiments involved a problem of limited size and complexity and a fixed server (and then a centralized architecture) has been used to store and analyze sensor data.. Extracting high-level information from the raw data captured by sensors and representing it in machine-understandable languages has several interesting applications [34,21]. The paper [14] surveyed requirements, solutions and challenges in the

[3]OWL 2 Web Ontology Language Document Overview (Second Edition), W3C Recommendation 11 December 2012, `http://www.w3.org/TR/owl2-overview/`

Table 1
Syntax and semantics of $\mathcal{ALN}$ constructs

Name	DL syntax	OWL RDF/XML element	Semantics
Top	$\top$	`<owl:Thing>`	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	`<owl:Nothing>`	$\emptyset$
Concept	C	`<owl:Class>`	C
Role	R	`<owl:ObjectProperty>`	C
Conjunction	$C \sqcap D$	`<owl:intersectionOf>`	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Atomic negation	$\neg A$	`<owl:disjointWith>`	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$
Unqualified existential restriction	$\exists R$	`<owl:someValuesFrom>`	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$
Universal restriction	$\forall R.C$	`<owl:allValuesFrom>`	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$
Unqualified number restrictions	$\geq nR$	`<owl:minCardinality>`	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$\leq nR$	`<owl:maxCardinality>`	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$
Definition axiom	$A \equiv C$	`<owl:equivalentClass>`	$A^{\mathcal{I}} = C^{\mathcal{I}}$
Inclusion axiom	$A \sqsubseteq C$	`<owl:subClassOf>`	$A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

area of information abstraction and presented an efficient workflow based on the current state of the art.

Semantic Web research has addressed the task of describing sensor and data features through ontologies; relevant collections are on the following ontology catalogues: Linked Open Vocabularies (LOV) [54], LOV for Internet of Things (LOV4IoT)[4], OpenSensingCity[5] and `smartcity.linkeddata.es`[6] developed within the *READY4SmartCities* project. Best practices and methodologies for integrating Semantic Web technologies and ontologies in the IoT are discussed in [17]. *SSN-XG*[7] [9] is perhaps the most relevant and widely accepted ontology in this field. It is general enough for being adapted to different applications. In addition, it is compatible with the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) standards at the sensor and observation levels [5]. OGC SWE has been used in several frameworks aiming to grant access to sensor data as RESTful services or Linked Data [23,11]. Many projects, *e.g.*, SPITFIRE [41], put semantics in networking protocols to build full communication frameworks. The problem of semantic data flow compression in scenarios with limited resources has been faced in [12] by developing a scalable middleware platform to publish annotated data streams on the Web through HTTP.

Unfortunately, the above solutions only allow elementary queries in SPARQL fragments on RDF annotations. More effective techniques such as ontology-based *Complex Event Processing* (CEP) [7] exploit a shared domain conceptualization to define events and actions to be run on an event processing engine. Also the *ENVISION* [29] and ETALIS [3] projects combine CEP with semantic technologies to perform Semantic Event Processing from different sources: context and background knowledge are represented in RDF, while SPARQL queries are exploited to identify complex event patterns from incoming facts populating a knowledge base. The *ACEIS* CEP middleware [15] processes urban data streams in smart city applications: a semantic information model has been designed to represent complex event services and it is leveraged for discovery and integration of sensor data streams via SPARQL queries. Nevertheless, a fundamental limitation of CEP approaches is that pattern detection relies on rigid Boolean outcomes of defined queries and rules, whereas a more flexible approximated match could better support classification. In fact, the adoption of KR techniques on large amounts of instances is useful to annotate raw data and produce high-level descriptions in a KB. This is suitable for advanced reasoning, aiming to improve standard data mining and ML algorithms [44]. In [32] a post-processing of ML operations –based on ontology consistency check– aims to improve results of association rule mining. Semantically inconsistent associations were pruned and filtered out, leveraging logic reasoning. The framework proposed in [60] combines ontology-based Linked Open Data (LOD) sources as background knowledge with dynamic sensor and so-

[4] `http://lov4iot.appspot.com/`
[5] `http://ci.emse.fr/opensensingcity/ns/ontologies/`
[6] `http://smartcity.linkeddata.es/`
[7] Semantic Sensor Network Ontology, W3C Recommendation 19 October 2017, `https://www.w3.org/TR/2017/REC-vocab-ssn-20171019`

cial data, in order to produce dynamic feature vectors for model training. Similarly, [59] exploits ontology-based data annotation to perform classification and resource retrieval. A LOD-inspired approach and an architecture are presented in [18] to define, share and retrieve rules for processing sensor data in the IoT. While the above works define from scratch complete semantic IoT platforms, the present paper focuses on a specific ML approach, which could be integrated in larger frameworks. Furthermore, the above proposals exploit SPARQL queries for reasoning and implementing rules on annotated data, while the approach proposed here aims to provide more thorough and robust answers, by supporting also non-exact matches via non-standard DL inferences. Extensions to standard reasoning algorithms, supporting uncertainty and time relationships, have been also proved as effective in tasks such as activity recognition [36].

Some of the most successful ML methods, such as ANN and *deep learning* techniques, suffer from opaqueness of models, which cannot be interpreted by human experts and therefore cannot explain reasons for the outcomes they provide. This is a serious issue for ML adoption in all those sectors which require accountability of decisions and robustness of outputs against accidental or adversarial input manipulation [24,37]. Research efforts to build decipherable results of ML techniques and systems are therefore more and more expanding. The approach adopted in this paper could be an example of that, as it combines semantic similarity measures and classical (frequentist) data stream mining [25]. Another conceptually easy approach is to exploit ensemble learning, by combining multiple low-dimensional submodels, where each individual submodel is simple enough to be verifiable by domain experts [37]. In [27] Bayesian learning is used to generate lists of rules in the *if* ... *then* form, which can provide readable reasons for their predictions. That method is competitive with state-of-the-art techniques in stroke prediction tasks over large datasets, although training time appears as rather long for IoT scenarios. The regression tool in [30] is able to translate automatically components of the model to natural-language descriptions of patterns in the data. It is based on a compositional grammar defined over a space of Gaussian regression models, which is searched greedily using marginal likelihood and the Bayesian Information Criterion (BIC). The approach supports variable dimensionality (number of features) in each regression model, thus allowing the selection of the desired trade-off between accuracy and ease of interpretation.

Semantic-enhanced ML methods can achieve the same goals through formal, logic-based descriptions of models and outputs in order to develop explanatory functionalities, so increasing users' trust. Furthermore, they can be integrated in larger cognitive systems, where models and predictions are used for automated reasoning. *Semantic data mining* refers to data mining tasks which systematically incorporate domain knowledge in the process. The framework proposed here belongs to this research area, which has been surveyed in [10,45] and includes ontology-based rule mining, classification and clustering. Ontologies are useful to bridge the semantic gap between raw data and applications, as well as to provide data mining algorithms with prior knowledge to guide the mining process or reduce the search space. They can be successfully used in all steps of a typical data mining workflow. In Ontology-Based Information Extraction (OBIE) [57], ontologies are also exploited to annotate the output of data mining, and this aspect is also adopted in our approach. In [53], the authors propose to use wireless sensor networks and ontologies to represent and infer knowledge about traffic conditions. Raw data are classified through an ANN and mapped to ontology classes for performing rule-based reasoning. In [8], an unsupervised model is used for classifying Web Service datatypes in a large number of ontology classes, by adopting an extended ANN. Also in that case, however, mining is exploited only to map data to a single class.

The exploitation of matchmaking through non-standard inferences enables a fine-grained event detection by treating the ML classification task as a resource discovery problem. Promising semantic-based approaches also include fuzzy DL learning [28], concept algebra [56] and tensor networks based on Real Logic [51]. While their prediction performance appears good, computational efficiency must be still evaluated completely and accurately, before considering them suitable for IoT scenarios.

Summarizing, most semantic-enhanced data mining and machine learning approaches currently support meaningful data description and interpretation, but provide limited reasoning capabilities and have complex architectures. Conversely, classical ML algorithms can achieve high prediction performance, but their models and outcomes often have poor explainability. This paper proposes an approach aiming to combine the benefits of Semantic Web technologies with state-of-the-art learning effectiveness, particularly for IoT data stream mining.

4. Automatic identification of events via knowledge representation

MAFALDA preserves the classical data mining and machine learning workflow: data collection and cleansing, model training, validation and system usage. Nevertheless, as reported in Figure 1, semantic enhancements grounded on DLs change the way each step is performed. Details about the devised methodology are outlined hereafter.

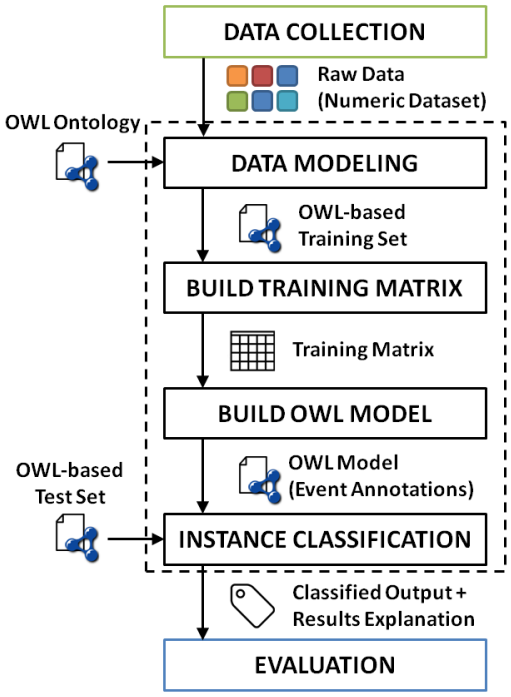


Fig. 1. Framework architecture

4.1. Ontology and data modeling

The overall workflow starts with raw data gathered *e.g.*, by sensors dipped in a given environment extracting several different parameters, *a.k.a. features*. In order to support semantic-based data annotation and interpretation, an ontology $\mathcal{T}$ models the domain conceptualization along *patterns* specified hereinafter. $\mathcal{T}$ is assumed as acyclic and expressed in the moderately expressive $\mathcal{ALN}$ DL. This is required by the non-standard inferences for semantic matchmaking [50] exploited in the subsequent stages. *M3-lite* [1] has been used as upper ontology. For each measuring parameter (*e.g.*, acceleration, engine load, fuel consumption), $\mathcal{T}$ must include a hierarchy of concepts (each one with its own properties) derived from the `QuantityKind` concept in M3-lite, forming a partonomy of the topmost concept. In other words, each parameter is represented via a class/subclass taxonomy featuring all significant value ranges and configurations it can assume in the domain of interest. The depth of the hierarchy and the breadth of each level will be chosen by the knowledge modeler; they are typically proportional to both resolution and range of sensing/capturing equipment, as well as to the needed degree of detail in data representation.

As an example, Figure 2 shows the class hierarchy of the domain ontology modeled for the case study in Section 5. Two different modeling approaches have been investigated to represent data ranges for measured parameters in the knowledge base[8]. In the first one, each data range associated to a concept is modeled by means of a pair of OWL *annotation properties*, named `maxValue` and `minValue`, indicating the maximum and minimum value, respectively. For example, the concept `EngineRPMLevel2` – corresponding to values from 801 to 900 engine revolutions per minute– is annotated as in Figure 3(a). Afterwards, this approach has been modified: the modeling preserves the hierarchy of concepts, but an explicit semantics is given to the range of potential variability for each measured parameter. *Number restrictions* associated to each subconcept have been adopted for that purpose. See the `EngineRPMLevel2` concept expressed in this way in Figure 3(b). The latter approach allows a semantic-based selection also in the preliminary step of raw data collection: numerical data are translated to number restrictions and the correct corresponding concept subclass is identified exactly by means of the *Consistency Check* reasoning service [50]. Performance differences related to the above modeling approaches are described in Section 6.

According to the proposed modeling, a generic data corpus can be translated to an OWL-based dataset, where each record corresponds to an instance of a proper KB. Regardless of the particular ontology modeling, each individual also includes a set of *annotation properties* setting up the real output class for each observable event. As shown in Figure 4, the *annotation property* name reflects the output attribute, while

[8]Both the proposed OWL ontologies are available on the project repository: http://github.com/sisinflab-swot/mafalda

Fig. 2. Class hierarchy in the domain ontology for the case study

the annotation value refers to the output concepts associated during the dataset building. In particular, the *Traffic Danger* ontology [55] has been exploited in the road monitoring case study described in Section 5 to model traffic congestion and road surface conditions, while novel classes have been defined to represent the different driving styles. In this way, both a single event annotation and the overall dataset –described w.r.t. well-known vocabularies– can be also: (i) published on the Web following the Linked Data guidelines [20]; (ii) shared with other users or IoT devices on the same network; (iii) reused locally for further reasoning and processing tasks. The modeling effort is basically a study and manual design procedure. As per many engineering activities, it could be supported by software tools for computer-aided design. Protégé by Stanford University[9] is one of the most adopted and widespread ontology editors and knowledge management systems; anyway, also more simplistic and easy-to-use software could be suitable in this case, given the fixed patterns to be followed. In particular, the proposed ontology modeling complies with different tools (*e.g.*, WebVOWL [31] and LODE [39]) aiming to improve visualization and automatic documentation of OWL ontologies.

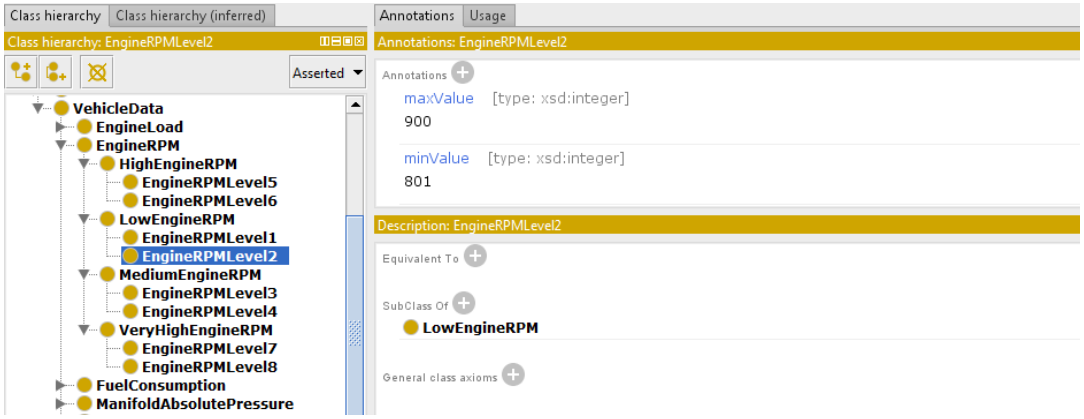
[9]https://protege.stanford.edu

4.2. Training

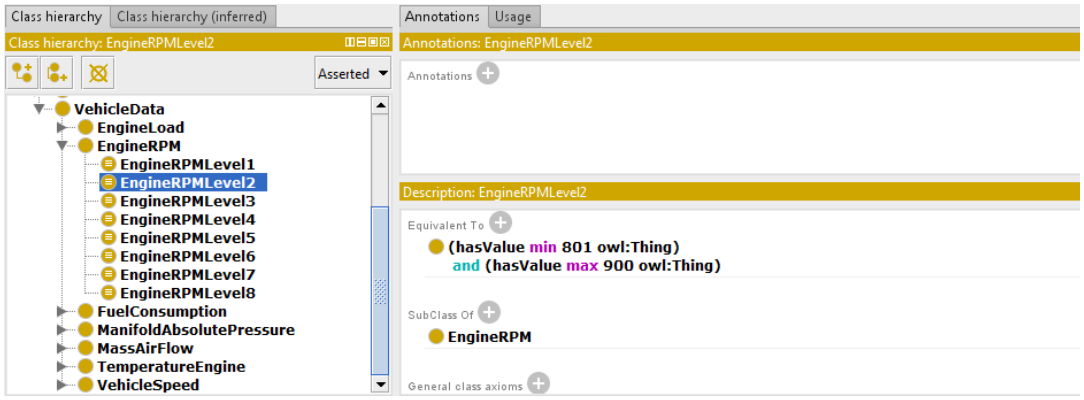
In this phase training data are automatically generated, able to define the *model* used afterward by the ML algorithm for predictions on test data. In the proposed approach, there is a semantic annotation for each possible output class, connoting the observed event/phenomenon according to input data. In these terms, the framework presents a twofold modeling effort: the Knowledge Base definition (which is basically human-driven, manually pursued) and the training set generation (automatically carried out after the first step). The annotations will be expressed in *Concept Components* according to the following recursive definition:

Definition 1 (Concept Component) *Let C be an $\mathcal{ALN}$ concept formalized as $C_1 \sqcap \cdots \sqcap C_m$. The* Concept Components *of C are defined as follows: if C_j, with $j = 1,\ldots,m$ is either a concept name, or a negated concept name, or a number restriction, then C_j is a concept component of C; if $C_j = \forall R.E$, with R $\mathcal{ALN}$ role and E $\mathcal{ALN}$ concept formalized as $E_1 \sqcap \cdots \sqcap E_p$, then $\forall R.E_h$ is a concept component of C, for each E_h concept component of E, $h = 1,\ldots,p$.*

The training phase is carried out on a set S of n training samples, each with at most m features. Let us suppose w distinct outputs exist in the training set and the system must be trained to recognize them. Each fea-



(a) Annotation properties



(b) Number restrictions

Fig. 3. Data range modeling

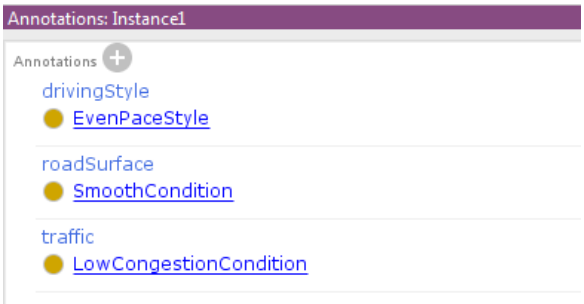


Fig. 4. Example of output class annotation

ture value is mapped to the most specific corresponding concept in the reference ontology $\mathcal{T}$. Therefore the i-th sample $\forall\, i = 1, \ldots, n$ is composed of: (a) up to m concept components $C_{i,1}, \ldots, C_{i,m}$ annotating its features; (b) an observed output O_i labeled with a class name in the ontology.

Samples are processed sequentially by Algorithm 1 in order to build the so-called *Training Matrix* $\mathcal{M}$ (the pseudocode uses a MATLAB-like notation for matrix access). $\mathcal{M}$ is a $(w+1) \times (k+1)$ matrix. All the different outputs are located on the first column while the k distinct concept components occurring in the training set are on the first row. In each element of the matrix, there is the number of occurrences of the column header concept component in the samples having the row header output. Basically, Algorithm 1 takes the i-th training sample and first checks its associate class O_i (lines 4-11): if it is not yet in $\mathcal{M}$ (no previous sample has been associated to that class), it appends a row to $\mathcal{M}$ setting its values to zeros. Subsequently, for each concept component $C_{i,j}$, if $C_{i,j}$ is not yet in $\mathcal{M}$ (*i.e.*, no previous sample includes that concept component), it appends a column and sets its values to zeros (lines 13-20). Finally, it increases by 1 the value of the cell corresponding to O_i and $C_{i,j}$ (line 21).

Algorithm 1 Creation of the Training Matrix

Input:
– w: number of output classes;
– n: number of instances in the training set;
– m_i: number of concepts defining the description of the instance $S_i \; \forall \; i = 1, \ldots, n$;
– $\mathcal{L}$: reference Description Logic;
– $\mathcal{T}$: acyclic TBox;
– O: vector of the output classes $O_1, O_2, \ldots, O_w$;
– S: training set containing $\{S_1, S_2, \ldots S_n\}$ instances, with $S_i = (C_{i,1}, \ldots, C_{i,m_i}, O_i) \; \forall \; i = 1, \ldots, n$, where $C_{i,j}$ concepts and O_i are expressed in $\mathcal{L}$ and satisfiable in $\mathcal{T}$.

Output:
– k: number of distinct concepts appearing in S;
– $\mathcal{M} : (w+1) \times (k+1)$ matrix of occurrences of the concept components for each observed output.

$\mathcal{M} := 0$ // start with a (1×1) matrix
$r := 1, c := 1$
for $i := 1$ to $|S|$ **do**
  $u_r :=$ *findConceptIndex*$(O_i, \mathcal{M}(:,1))$
  if $u_r =$ *null* **then**
    append a row to $\mathcal{M}$
    $r := r + 1$
    $u_r := r$
    $\mathcal{M}(u_r, 1) := O_i$
    set $\mathcal{M}(u_r, 2 : c)$ to zero
  end if
  for $j := 1$ to m **do**
    $u_c :=$ *findConceptIndex*$(C_{i,j}, \mathcal{M}(1,:))$
    if $u_c =$ *null* **then**
      append a column to $\mathcal{M}$
      $c := c + 1$
      $u_c := c$
      $\mathcal{M}(1, u_c) := C_{i,j}$
      initialize $\mathcal{M}(2 : r, u_c)$ to zeros
    end if
    $\mathcal{M}(u_r, u_c) = \mathcal{M}(u_r, u_c) + 1$ // update occurrences
  end for
end for
return k, $\mathcal{M}$

$\mathcal{M}$ gives a complete picture of the training set. Each output class can be defined now as conjunction of the concepts having greater-than-zero occurrences in the corresponding row. By doing so, however, even very rare concept components are included, which may have low significance in representing the class. Therefore, it is useful to define a *significance threshold* T_s as the minimum number of samples a concept component must appear in, to be considered relevant for the occurrence of a particular output. The structure of $\mathcal{M}$ suggests the possibility to define different thresholds for each output and for each feature:

$$T_{s(i,j)} = \theta_{(i,j)} \, |S|$$

with $0 < \theta_{(i,j)} \leq 1 \;\; \forall i, j$ being adaptive ratios computed through *e.g.*, a cross-validation process on the training dataset.

Customized thresholds allow to focus sensitivity on the features with highest variance and/or the outputs most difficult to predict. In the road monitoring case study described in Section 5, the threshold value has been calculated in a way not disadvantaging sensors with lower sampling rate or events which occur less often in the training dataset. In detail, each element $\mathcal{M}$(i,j) is normalized: (i) according to the individual feature w.r.t. all the features belonging to the same class hierarchy (*i.e.*, all classes annotating *e.g.*, temperature value ranges) and (ii) based on a single event with respect to the remaining ones (*e.g.*, if "Uneven Road Condition" is much less frequent than "Smooth Road Condition", normalization will increase all the values in the "Uneven Road Condition" row of $\mathcal{M}$).

The adopted formula is:

$$T_{s(i,j)} = T_{base} * \frac{max_{occur}(i,j) - min_{occur}(i,j)}{2}$$

where T_{base} is a user-defined base percentage threshold. The result of the training is the association of every output class label O_i with a conjunctive expression composed by the concepts occurring with a normalized frequency over the threshold.

Therefore, this training approach produces a KB with conceptual knowledge (the TBox) modeled –as said– by human experts and factual knowledge (the ABox) created automatically from the available data stream, with instances representing the events the system is able to recognize.

4.3. Classification

This task refers to the typical ML problem of assigning each input instance to a possible output class, based on its features. The classification exploits a semantic matchmaking process based on *Concept Contraction* and *Concept Abduction* non-standard inference services [50].

Given an ontology $\mathcal{T}$ and two concept expressions A and B, if they have conflicting characteristics, Concept Contraction determines a concept expression G (*Give up*) which is an explanation about what in A is not compatible with B and returns a value $penalty_{(c)}$ representing the semantic distance associated to it. Otherwise, if A is compatible with B, but does not cover it fully, Concept Abduction calculates a concept

expression H (*Hypothesis*) representing what should be hypothesized (*i.e.*, is underspecified) in B in order to completely satisfy A, and it provides a related $penalty_{(a)}$ value. Concept Contraction and Concept Abduction can be respectively considered as extensions to *Satisfiability* and *Subsumption* standard inference services, which only provide "yes/no" answers in KR systems.

MAFALDA first labels instance elements to be classified with respect to the reference ontology, like in Section 4.1. Their conjunction is then taken as annotation of the instance itself. A linear combination of the penalty values obtained from matchmaking yields the *semantic distance* between the input instance and each event description O_i generated during training. In particular, based on the different ontology modeling techniques proposed in Section 4.1, two semantic distance functions have been defined. In the case of *annotation properties*, penalty score is computed via the following formula:

$$SD_{ap}(R,S) = \frac{penalty_{(a)}(R,S)}{penalty_{(a)}(R,\top)}$$

where $penalty_{(a)}(R,S)$ measures the Abduction-induced distance between an event description R and sensor data annotation S; this value is normalized dividing by the distance between R and the universal concept $\top$ which depends only on axioms in the ontology. Instead, in case of adopting *number restrictions*, the penalty function is defined as:

$$SD_{nr}(R,S) = \frac{\alpha * penalty_{(c)}(R,S) + \beta * penalty_{(a)}(R,S)}{penalty_{(a)}(R,\top)}$$

where $penalty_{(c)}(R,S)$ indicates the Contraction-induced semantic distance. This value is now present because *number restrictions* introduce explicit incompatibilities between concepts due to disjoint numeric ranges. Two tunable weighting factors combine both contributions and enable a ranking mainly based on either conflict or missing features.

After the penalty scores calculation, the predicted/recognized event will be the one with the lowest distance. Since semantic matchmaking associates a logic-based explanation to ranked (dis)similarity measures, the classification outcome has a formally grounded and understandable confidence value. This is a fundamental benefit with respect to the majority of standard ML techniques, which produce a prediction not simply intelligible. Furthermore, notice that the proposed approach does not take the instance annotation directly as output, because the inherent data volatility in IoT contexts could lead to inconsistent assertions, which would be impossible to reason on.

4.4. Evaluation

The system evaluation works with a test set, consisting of several classified instances referred to the same ontology used for building the training set. The goal is to check how often (and possibly, how much) the predicted event classes correspond to the actual events associated to each instance of the test set. Beyond classical performance indicators for classifying ML algorithms (like the *confusion matrix* and statistical metrics as accuracy, precision and recall), the graded nature of predictions of *MAFALDA*, *e.g.*, the average semantic distance of the predicted class from the actual one, allows applying typical error measures of regression analysis like the Root Mean Square Error (RMSE).

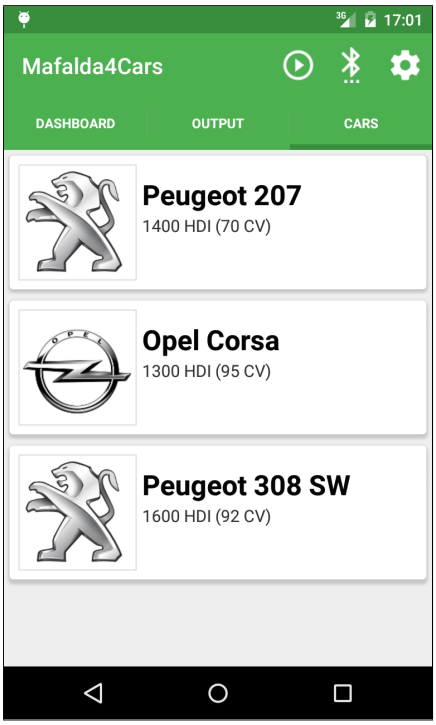
Cross-validation can be used to tune system parameters if performance is not satisfactory. Moreover, if computing resources permit it, incoming test data can also be used to update the training matrix on-the-fly, in order to allow the model to evolve when new data is observed.

5. Case study: road and traffic monitoring

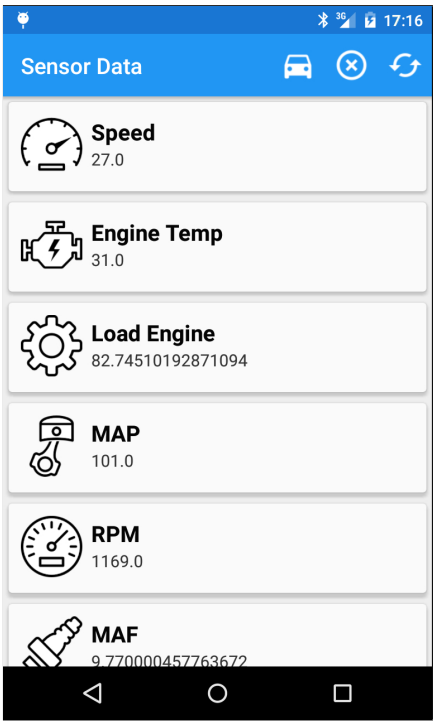
Mobility services are one of the main IoT application areas. The presented case study refers to a prototypical system for road and traffic monitoring created *e.g.*, to improve the functionality of navigation systems with real-time driver assistance. Useful insight on travel conditions is provided both among nearby vehicles (in a peer-to-peer fashion through VANETs – Vehicular Ad-hot NETworks) and on a large scale (*e.g.*, by updating a remote Geographical Information System with real-time and history information toward road policy makers). In particular, the proposed knowledge-based system exploits the semantic descriptions of vehicles and context annotations to:

1. interpret vehicle data extracted via the mandatory On-Board Diagnostics[10] (OBD-II) port;
2. integrate environmental information;
3. detect potential risk factors.

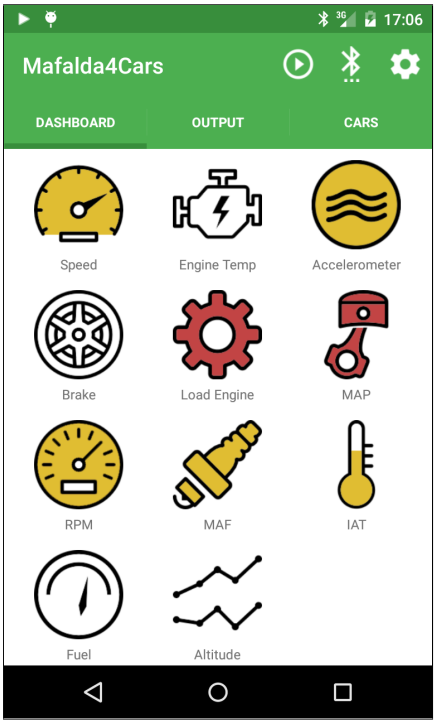
[10] California Environmental Protection Agency, On-Board Diagnostics (OBD) Program, `http://www.arb.ca.gov/msprog/obdprog/obdprog.htm`



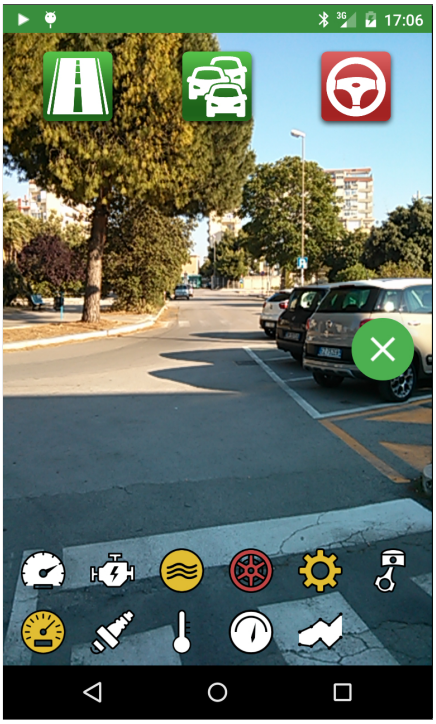
(a) Cars list



(b) Sensed data from OBD-II



(c) Measurements dashboard



(d) Classification view

Fig. 5. Mobile application screenshots

Besides providing warnings, the detected knowledge allows giving suggestions to the driver and evaluating car efficiency and environmental impact in real time [47].

The implementation is based on the Java language in order to be compatible with both Java SE (Standard Edition) and Android platforms. The prototype includes the *Mini-ME* lightweight matchmaker [50], which provides the required inferences for the $\mathcal{ALN}$ DL (under the assumption of acyclic TBoxes). The above ML framework has been used to extract high-level indications starting from a large number of low-level parameters acquired from the car via OBD-II and from micro-devices (accelerometer, gyroscope, GPS) embedded in the user's smartphone, with the goal of characterizing accurately the overall system composed by driver, vehicle and environment.

A dedicated dataset has been collected for experimental analyses[11]. Raw data have been retrieved and stored using the *Torque Lite (OBD-II & Car)*[12] Android application on seven different routes: suburban, urban and mixed ones, with medium and long distances. An average of five traces per route have been recorded, sampling OBD-II parameters and smartphone data at 1 Hz frequency. About 10,000 records have been collected on average for each route, taken on different days, in various traffic conditions and with three different cars (and drivers): particularly a Peugeot 207 (two routes), an Opel Corsa (two routes) and a Peugeot 308 (three routes) have been used.

The case study aims to identify driving style as well as road characteristics and traffic conditions, by analyzing parameters gathered by the car and by the user's smartphone. It is purposely kept simple to give an immediate proof of concept, but classification can be largely enriched at will without modifying the theoretical settings. In detail, the system should detect the following classes:

- *Smooth*, *Uneven* or *FullOfHoles* road surface conditions;
- *Low*, *Normal* or *High* traffic congestion conditions;
- *Aggressive* or *EvenPace* driving style.

During the dataset creation, each driver who collected a trace has been asked to label manually the records with the event characteristic for each of the above categories. Gathered information represent the raw data in the ML problem. Timestamp and GPS coordinates (also taken through the smartphone) have been added to each record.

Analyzed data consist of:

- *altitude change*, calculated over 10 seconds;
- *speed*: current value, average and variance in the last 60 seconds and change in speed for every second of detection;
- *longitudinal and vertical acceleration*, measured by the smartphone accelerometer and pre-processed with a low-pass filter to delete high frequency signal components due to electrical noise and external forces;
- *engine load*, expressed as percentage;
- *engine coolant temperature*, in °C;
- *Manifold Air Pressure* (MAP), a parameter the internal combustion engine uses to compute the optimal air/fuel ratio;
- *Mass Air Flow* (MAF) *Rate* measured in g/s, used by the engine to set fuel delivery and spark timing;
- *Intake Air Temperature* (IAT) at the engine entrance;
- *Revolutions Per Minute* (RPM) of the engine;
- *average fuel consumption* calculated as needed liters per 100 km.

As shown in Figure 2, the above parameters have been represented in the domain ontology and divided in subclasses, each characterized by a value range. At the end of the training phase, the ABox created automatically from the available data stream contains instances representing the events that the system should be able to recognize.

In addition to evaluations on the static data set, a mobile application for smartphones has been developed to validate the framework in a real usage. It is an evolution of [47], devoted to evaluate vehicle health and driver risk level, exploiting semantic-based matchmaking to suggest users how to reduce or even eliminate danger and get better vehicle performance and lower environmental impact. By exploiting this new version, implemented using Android SDK Tools, Revision 24.1.2 –corresponding to Android Platform version 5.1, API level 22– and tested on a LG E960 *Nexus 4* smartphone, the user can:

- select a dataset related to the cars used in the experiments (Figure 5(a)) and train the prediction model;

[11] A subset of the collected data is publicly available on the project GitHub repository cited in Section 4.1.

[12] http://torque-bhp.com/

– view and query all available sensed data, as shown in Figure 5(b);
– open a measurements dashboard (see the screenshot in Figure 5(c)). For each device, a colored icon indicates a low (white), medium (yellow) or high (red) measured value.

Moreover, the user can start the classification view in Figure 5(d). The smartphone camera viewfinder is used as background allowing the user to see the classification outputs without looking away from the road. The application queries vehicle information via OBD-II and executes the algorithm described in Section 4. The user interface shows at the bottom a compact device dashboard (a smaller version of the one in Figure 5(c)), while three large icons are displayed at the top, related to the event outputs (road conditions, traffic and driving style). Also in this case, classified output levels correspond to different colors (green, yellow and red). In the picture, the algorithm detects a smooth road and low traffic (green icons) and an aggressive driving style by the user (red icon).

6. Experiments

This section reports on the experiments carried out on the dataset collected as stated before. Results are summarized and compared through classic ML metrics such as weighted precision, recall, F-score and overall accuracy.

6.1. Configuration selection

A preliminary test (Table 2) compares performance indexes of the modeling techniques described in Section 4.1 with data ranges expressed through *number restrictions* (NR_i) and *annotation properties* (AP_j), respectively. For each route, the whole dataset has been split in a training set and a test set by holdout, mixing the records randomly in a 70%/30% ratio. The training set generates the model, while the test set allows evaluating the classification performance. Training and test set have been processed in several configurations obtained by varying T_{base}, *i.e.*, the normalization threshold value, as well as α and β, used to compute the semantic distance in the classification task. For each test configuration, performance measures have been calculated.

Precision and recall values are plotted in Figure 6. The best configuration is AP1, presenting the highest values for recall, F-score and accuracy; precision is only slightly lower than configurations with larger T_{base}. It is important to notice that configurations including *number restrictions* exhibit lower values due to the disjunction of intervals in modeling concepts of the ontology: semantic descriptions produced by the training stage are all similar, penalizing the later stages. Indeed, in the classification phase, the matchmaking between the output description generated by the training and the sample from the test set tends to increase penalty values due to disjoint number restrictions, frequently producing an incorrect classification output.

6.2. Performance comparison

The same training and test sets have been used with classical Machine Learning algorithms to compare and evaluate results obtained with the best configuration of *MAFALDA*. The following algorithms recalled in Section 3.1 have been used for comparison:

1. J48 implementation of C4.5;
2. Functional Tree (FT);
3. Random Tree (RT);
4. K-Nearest Neighbors (k-NN);
5. Multilayer Perceptron;
6. Deep Neural Network (DNN) Classifier.

Algorithms 1-5 have been tested in their implementation from *Weka*[13] [19]; the last one is implemented in the `tf.estimator.DNNClassifier` class of *TensorFlow*[14]. Also in this case, each algorithm is used for testing different configurations obtained by conveniently setting parameters in Table 3. Results corresponding to the configurations with the highest accuracy are reported in Table 4. *MAFALDA* presents comparable precision, albeit with slightly lower recall values. Overall, it represents a competitive alternative to classical ML algorithms, with the benefit of producing interpretable semantic-based annotated concept representations.

Moreover, the experimental analysis has measured processing time of compared algorithms on a PC testbed, equipped with Intel Core i7-3770K CPU at 3.5 GHz, 12 GB DDR3 SDRAM memory, 2 TB SATA (7200 RPM) hard disk, 64-bit Microsoft Windows 7 Professional, 64-bit Java 8 SE Runtime Environment build 1.8.0_31-b13, and 64-bit Python 3.6.3 environment. Training and evaluation times, reported in Ta-

[13] Weka version 3.6.12, http://www.cs.waikato.ac.nz/ml/weka/
[14] TensorFlow version 1.4.0, http://www.tensorflow.org/

Table 2
Experiments report in several different test configurations

	ID	α	β	T_{base}	Precision	Recall	F-Score	Accuracy
KB with Number Restrictions	NR1	0.2	0.8	50	0.741	0.575	0.648	0.575
	NR2	0.5	0.5	50	0.846	0.661	0.709	0.661
	NR3	0.2	0.8	20	0.742	0.609	0.669	0.609
	NR4	0.5	0.5	20	**0.890**	0.672	0.766	0.672
KB with Annotation Properties	AP1	-	-	15	0.861	**0.813**	**0.836**	**0.813**
	AP2	-	-	20	0.866	0.798	0.831	0.798
	AP3	-	-	30	0.867	0.764	0.812	0.764
	AP4	-	-	50	0.866	0.665	0.752	0.665
	AP5	-	-	65	0.837	0.624	0.715	0.624

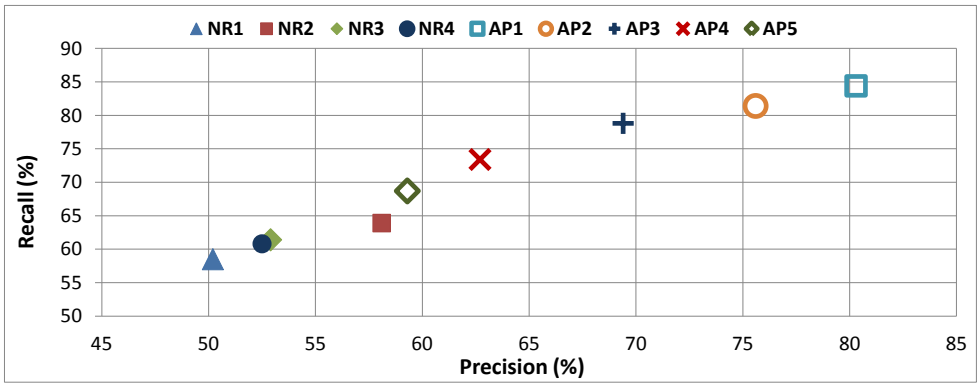


Fig. 6. Precision/recall plot

ble 4, represent the average interval (computed on five runs) needed to build the model –starting from each training set exploited for accuracy analysis– and perform the evaluation on the related test set. The highest training time has been taken by the DNN Classifier, due to the complex model and the expensive optimization function, whereas the lowest is by k-NN, where the training task only validates input data. Conversely, the evaluation time is highest in the case of k-NN, since the algorithm calculates the distance among samples. Random Tree has the lowest overall time, due to the very simple model used to classify the test instances. MAFALDA exhibits a very low training time, making the approach suitable for on-the-fly data stream processing, while evaluation time is higher due to semantic matchmaking. The above behavior gives however a satisfactory performance tradeoff in case of mobile ad-hoc scenarios as they are typically characterized by data rates higher than query rates.

Processing time of *MAFALDA* has been analyzed on two more platforms:

- *Nexus 4* smartphone, equipped with Qualcomm Snapdragon S4 quad-core CPU at 1.5 GHz, 2 GB RAM and Android 5.1.1 operating system;
- *Raspberry Pi Model B*[15], equipped with a single-core ARM11 CPU at 700 MHz, 512 MB RAM (shared with GPU), 8 GB storage memory on SD card, Raspbian Wheezy OS.

In this case, the tests have been executed using the first Peugeot 207 dataset consisting of 8615 records; 6030 used as training set and 2585 as test set. Like in the above experiment, each test has been repeated five times and the average value has been taken, as reported in Figure 7.

The overall process includes several sub-steps:

1. *Ontology Loading*: the OWL file containing the TBox $\mathcal{T}$ is loaded and parsed;
2. *Data Mapping*: for each of the 6030 data records in the training set, the concept subclass(es) corresponding to the parameter values are identified;

[15] http://www.raspberrypi.org/products/model-b/

Table 3
Parameters of the reference classification algorithms

Algorithm	Parameter	Description
J48	-M 2	minimum number of instances per leaf
	-U	unpruned tree
Functional Tree	-I 20	fixed number of iterations
	-F 0	tree type to be generated
	-M 20	minimum number of instances for node split
	-W 0	value for weight trimming
Random Tree	-K 0	number of attributes to randomly investigate
	-M 1.0	minimum number of instances per leaf
	-S 1	seed for random number generator
k-Nearest Neighbors	-K 1	number of nearest neighbors (k)
	-W 0	maximum number of training instances maintained
	-A LinearNNSearch	nearest neighbour search algorithm to use
Multilayer Perceptron	-N 50	number of epochs to train through
	-H 4,8,4	number of nodes on each hidden layer
DNN Classifier	num_epochs = 2	number of epochs to train through
	hidden_units = $[4, 8, 4]$	number of nodes on each hidden layer
	optimizer = 'Adagrad'	optimization algorithm to train the model
	activation_fn = tf.nn.relu	activation function of nodes

Table 4
Comparison of ML algorithms

Algorithm	Precision	Recall	F-Score	Accuracy	Training Time (ms)	Evaluation Time (ms)
J48	0.885	**0.883**	**0.884**	**0.883**	45.64	1.32
Functional Tree	0.884	0.880	0.882	0.876	565.52	165.88
Random Tree	0.879	0.879	0.879	0.879	14.87	**0.86**
k-Nearest Neighbors	0.878	0.863	0.870	0.863	**1.25**	914.71
Multilayer Perceptron	0.853	0.860	0.856	0.860	873.95	5.72
DNN Classifier	**0.905**	0.805	0.850	0.856	9898.05	430.67
MAFALDA	0.861	0.813	0.836	0.813	13.63	190.97

3. *Matrix Creation*: the Training Matrix using the concepts detected in the previous step is created/updated;
4. *Matrix Normalization*: this activity normalizes the matrix values and calculates the reference thresholds;
5. *OWL Model Creation*: starting from the normalized matrix, the semantic annotations describing each event are generated;
6. *Classification*: every data record is classified in the test set.

On both platforms, processing times obtained with a KB modeled with annotation properties are slightly faster than those obtained with number restrictions. Data mapping is by far the longest phase, due to the large amount of sensed data to manage. However, the time needed for a single mapping is very low (shorter than 1.2 ms on PC, 48 ms on smartphone and 70 ms on Raspberry). In case of number restrictions, ontology loading and data mapping are slower, due to the higher time needed to parse these kind of logic descriptions. Conversely, OWL model creation is faster because event annotations usually contain less concepts. In fact, for each parameter at most one subclass is associated to the event annotation given the explicit incompatibility among concepts induced by the number restrictions.

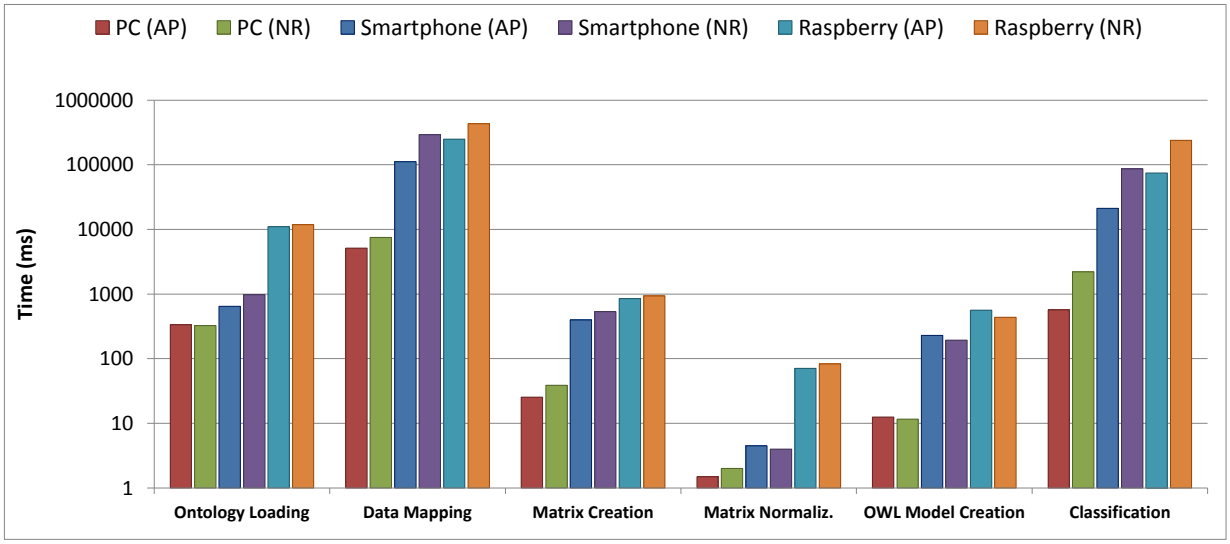


Fig. 7. Processing Time

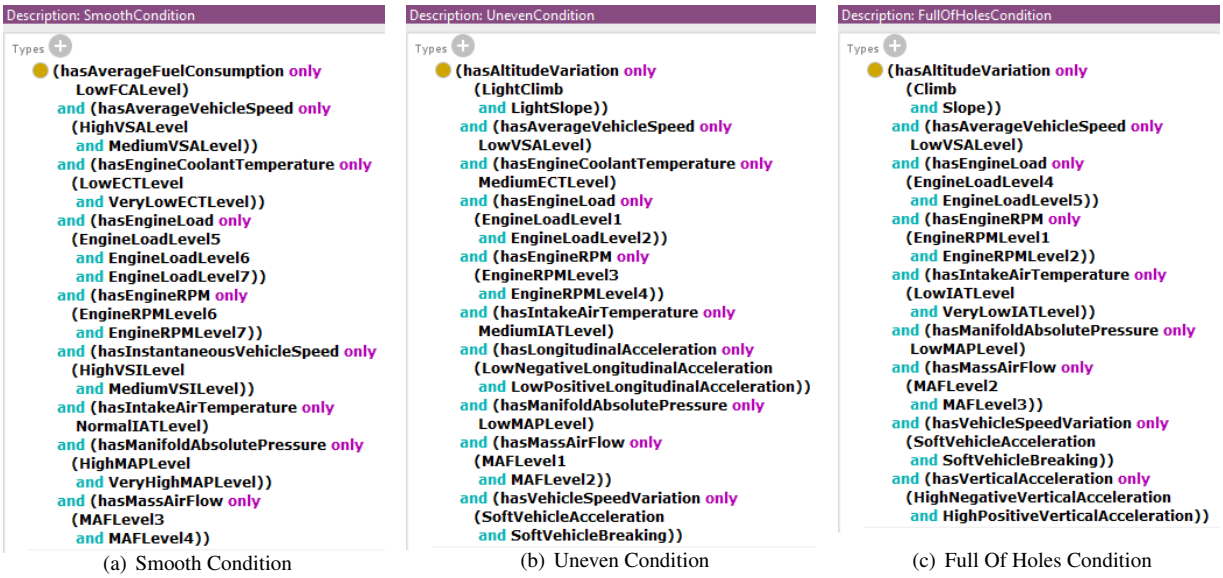
Considering the faster approach with annotation properties, the average turnaround time for training the classification model (*i.e.*, build and normalize the training matrix and then create the event annotations) is 40 ms on PC, 630 ms on smartphone and 1.48 s on Raspberry. This can be deemed as acceptable also for mobile and embedded systems. It is useful to point out that model training is performed only once after training set selection. Furthermore, processing time is clearly negligible with respect to data gathering: in the tested case, 6030 records read at 1 Hz frequency correspond to over 1.5 hours of data collection. Then, the classification task starts. For each test sample, classification is executed in about 0.22 ms on PC, 8 ms on mobile and 29 ms on Raspberry.

6.3. Explainability

In addition to adequate prediction performance and computational sustainability, a major goal of the proposed approach is to improve explainability w.r.t. the state-of-the-art techniques. Models and outputs generated by all the above algorithms applied to the first Peugeot 207 training set have been compared in a qualitative assessment. Figure 8 shows the model produced by MAFALDA. Classes are not simply labeled, but they are annotated as described in Section 4.2. Annotations are machine-understandable and easily readable. As said, the further semantic matchmaking enables also logic-based explanation of results: this grants accountability for classification outcomes. Finally, non-monotonic inference services and approximated matches increase resilience against missing or spurious sensor readings in test samples during system usage.

Figure 9 shows the model produced by the C4.5 decision tree (for the sake of readability, the figure reports only on a portion of the whole model). Every node tests a feature with a threshold: branches define paths leading to leaf nodes, which represent classification decisions. Also in this case, the model is easily readable both by humans and software systems: every class is basically represented by a clause in Disjunctive Normal Form. Nevertheless, the use of sharp thresholds in propositional atoms may make the approach vulnerable to slight sample data variations, *e.g.*, due to measurement problems. Functional Tree and Random Tree models –not reported for the sake of conciseness– are similarly readable, although nodes contain logistic or linear functions, respectively, instead of Boolean propositions. Due to the same reason, however, they are more robust against input perturbation.

Models generated by k-NN basically consist in point clouds –where each element represents a training sample– in an n-dimensional space, if n is the number of features. k-NN extensions toward hierarchical multi-label classification [42] have been proposed to predict structured outputs, but they are applicable only to multi-label classification problems, *i.e.*, when each sample can be labeled as belonging to multiple classes simultaneously; this is unlike most IoT and data stream mining scenarios.



(a) Smooth Condition (b) Uneven Condition (c) Full Of Holes Condition

Fig. 8. Example of the model generated by MAFALDA for road surface

Finally, the model generated by Multilayer Perceptron is depicted in Figure 10: input features are in green, output classes in yellow, neurons are organized in layers and their connections are shown. The model for the DNN Classifier is structurally similar. Such models are practically black boxes, both for humans and automatic systems, because the relationship between input sample features and output class label is encoded only in node thresholds and edge weights. For example, the first node of the second hidden layer is modeled as $x_{2,1} = f(\sum_{j=1}^{4} w_{1,j,1} x_{1,j}, t_{2,1})$ where f is the *activation function*, parameterized by the node threshold value $t_{2,1}$. Consequently, a meaningful description cannot be associated to output class labels. Furthermore, deep ANNs are particularly vulnerable to input perturbation and adversarial examples [38]: small variations in input data can produce widely different outputs in unpredictable ways. This lack of robustness and accountability prevents more widespread DNN adoption in the industry. Significant research efforts are ongoing to devise new approaches which are both more robust and explainable. Other state-of-the-art ML algorithms like *Support Vector Machines* are more resilient against input variations, but have similar model explainability issues.

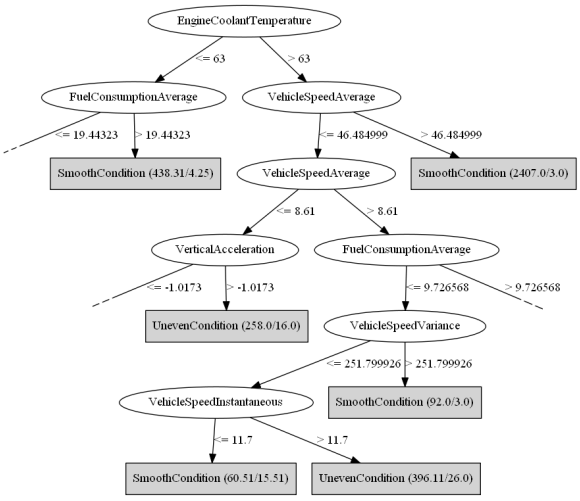


Fig. 9. Example of model for C4.5 decision tree classifier

6.4. Discussion

The main outcomes of experiments are summarized hereafter:

- In the reference road and traffic analysis case study, the best algorithm performance has been achieved using annotation properties and a low base threshold value. As number restrictions on disjoint value ranges cause more frequent inconsistency between output classed and test samples, semantic penalties become higher, leading to lower precision and recall.
- Prediction performance of MAFALDA is acceptable w.r.t. classical ML algorithms for IoT scenar-

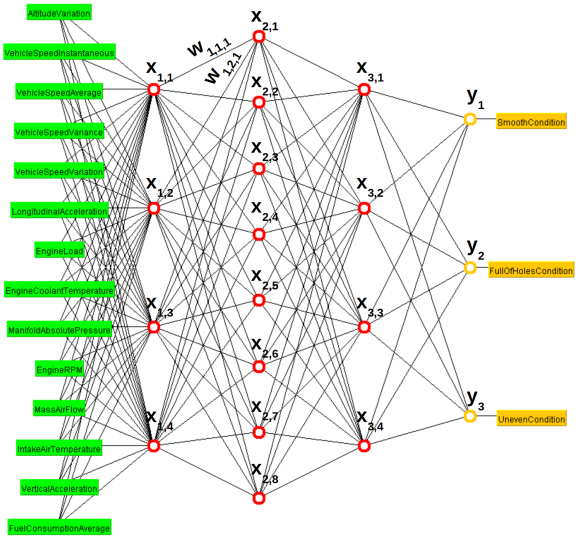


Fig. 10. Example of model for Multilayer Perceptron classifier

ios. Precision is comparable and recall is slightly lower in the case study.

– Processing time of MAFALDA is in the same order of magnitude as other ML algorithms. It has very fast model training and relatively slow evaluation time (though absolute values appear as adequate, with 8 ms per classification sample on the 2013 mobile device exploited in the case study). Hence, the semantic matchmaking induces a slowing down of the evaluation w.r.t. training. Anyway, in typical wireless sensor network and IoT scenarios where queries are less granular than input data streams, this however achieves satisfactory performances.
– Computational performance trends are predictable on PC vs mobile vs single-board computer platforms. Even for the latter, absolute values of training and classification times are small w.r.t. typical IoT application requirements. This is a significant outcome because it suggests that the proposed approach is responsive even with multiple features.
– MAFALDA achieves high explainability w.r.t. the state-of-the-art techniques. Other ML algorithms, such as decision trees, produce models with arguably similar or better readability for humans; notwithstanding, the proposed approach allows both formal semantic-based representation of the trained models and automatic logic explanation of classification outcomes. This makes the approach dependable and accountable, facilitating adoption even in critical scenarios. Moreover, MAFALDA output is expressed in standard Semantic Web languages, therefore it can be immediately used for further reasoning tasks. This facilitates integration in larger knowledge-based system architectures and is currently not allowed by other competitor approaches.

7. Conclusion and future work

This paper has introduced a novel approach for semantic-enhanced machine learning on heterogeneous data streams in the Internet of Things. Mapping raw data to ontology-based concept labels provides a low-level semantic interpretation of the statistical distribution of information, while the conjunctive aggregation of concept components allows building automatically a rich and meaningful representation of events during the model training phase. Finally, the exploitation of non-standard inferences for matchmaking enables a fine-grained event detection by treating the ML classification problem as a resource discovery.

A concrete case study on driving assistance has been developed through data gathered from real vehicles via On-Board Diagnostics protocol (OBD-II) and exploiting sensing micro-devices embedded on users' smartphones. A realistic dataset has been so built for experimentation. Subsequent extensive evaluations have allowed assessing performance of the proposed framework compared with state-of-the-art ML technologies, in order to highlight benefits and limits of the proposal. Main highlights are competitive prediction performance and speed w.r.t. existing approaches, combined with more expressive classification outputs and easily understandable models. In general, the main benefit of using Semantic Web technologies is to get meaningful information from data, but the main drawback is higher processing time: this paper demonstrates it is not necessarily true.

Several future perspectives are open for semantic-enhanced ML and particularly for the devised framework. A proper extension of the baseline training algorithm can enable a continuously evolving model through a fading mechanism allowing the system to "forget" the oldest training samples. A further extension of the training algorithm could aim at distributing the processing on more than one node, with a final merging step. This should reduce the communication overhead within a sensor network if intermediate nodes have enough storage capacity. Further vari-

ants could increase the flexibility of the proposed approach at the classification stage. For example, it could be useful to investigate the possibility of creating dynamically super-classes with a range combining those of the concepts found in the description: this would avoid affecting the result of the inference algorithms for descriptions that would otherwise be similar. Finally, adopting a more expressive logic language such as $\mathcal{ALN}(\mathrm{D})$ to model the domain ontologies could allow introducing data-type properties to better characterize typical IoT data features. Further experiments will have to be carried out to assess and optimize the proposed methods in terms of both accuracy and resource efficiency.

References

[1] R. Agarwal, D. G. Fernandez, T. Elsaleh, A. Gyrard, J. Lanza, L. Sanchez, N. Georgantas, and V. Issarny. Unified IoT ontology to enable interoperability and federation of testbeds. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 70–75, Dec 2016.

[2] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[3] D. Anicic, S. Rudolph, P. Fodor, and N. Stojanovic. Stream reasoning and complex event processing in ETALIS. *Semantic Web*, 3(4):397–407, 2012.

[4] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.

[5] M. Botts, G. Percivall, C. Reed, and J. Davidson. OGC® sensor web enablement: Overview and high level architecture. In *GeoSensor networks*, pages 175–190. Springer, 2008.

[6] R. Brachman and H. Levesque. The Tractability of Subsumption in Frame-based Description Languages. In *4th National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37. Morgan Kaufmann, 1984.

[7] K. Cao, Y. Wang, and F. Wang. Context-aware Distributed Complex Event Processing Method for Event Cloud in Internet of Things. *Advances in Information Sciences & Service Sciences*, 5(8):1212–1222, 2013.

[8] E. S. Chifu and I. A. Letia. Unsupervised semantic annotation of Web service datatypes. In *Intelligent Computer Communication and Processing (ICCP), 2010 IEEE International Conference on*, pages 43–50. IEEE, 2010.

[9] M. Compton, P. Barnaghi, L. Bermudez, R. García-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al. The SSN ontology of the W3C semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, Dec. 2012.

[10] D. Dou, H. Wang, and H. Liu. Semantic data mining: A survey of ontology-based approaches. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 244–251. IEEE, 2015.

[11] M. Fazio and A. Puliafito. Cloud4sens: a cloud-based architecture for sensor controlling and monitoring. *Communications Magazine, IEEE*, 53(3):41–47, 2015.

[12] J. A. Fisteus, N. F. García, L. S. Fernández, and D. Fuentes-Lorenzo. Ztreamy: A middleware for publishing semantic streams on the web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 25:16–23, 2014.

[13] J. Gama. Functional trees. *Machine Learning*, 55(3):219–250, 2004.

[14] F. Ganz, D. Puschmann, P. Barnaghi, and F. Carrez. A practical evaluation of information processing and abstraction techniques for the Internet of Things. *IEEE Internet of Things journal*, 2(4):340–354, 2015.

[15] F. Gao, M. I. Ali, E. Curry, and A. Mileo. Automated discovery and integration of semantic urban data streams: The ACEIS middleware. *Future Generation Computer Systems*, 76:561–581, 2017.

[16] A. Gyrard, C. Bonnet, K. Boudaoud, and M. Serrano. Assisting IoT projects and developers in designing interoperable Semantic Web of Things applications. In *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on*, pages 659–666. IEEE, 2015.

[17] A. Gyrard, M. Serrano, and G. A. Atemezing. Semantic Web methodologies, best practices and ontology engineering applied to Internet of Things. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 412–417. IEEE, 2015.

[18] A. Gyrard, M. Serrano, J. B. Jares, S. K. Datta, and M. I. Ali. Sensor-based Linked Open Rules (S-LOR): An Automated Rule Discovery Approach for IoT Applications and its use in Smart Cities. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 1153–1159. International World Wide Web Conferences Steering Committee, 2017.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

[20] T. Heath and C. Bizer. Linked data: Evolving the web into a global data space. *Synthesis lectures on the semantic web: theory and technology*, 1(1):1–136, 2011.

[21] C. A. Henson. *A semantics-based approach to machine perception*. PhD thesis, Wright State University, 2013.

[22] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[23] K. Janowicz, A. Bröring, C. Stasch, S. Schade, T. Everding, and A. Llaves. A restful proxy and data model for linked sensor data. *International Journal of Digital Earth*, 6(3):233–254, 2013.

[24] M. Jordan and T. Mitchell. Machine learning: Trends, perspectives, and prospects. *Clin. Pharmacol. Ther*, 349(6245):255–260, 2015.

[25] G. Krempl, I. Žliobaite, D. Brzeziński, E. Hüllermeier, M. Last, V. Lemaire, T. Noack, A. Shaker, S. Sievi, M. Spiliopoulou, et al. Open challenges for data stream mining research. *ACM SIGKDD explorations newsletter*, 16(1):1–10, 2014.

[26] N. Landwehr, M. A. Hall, and E. Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

[27] B. Letham, C. Rudin, T. H. McCormick, D. Madigan, et al. Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[28] F. A. Lisi and U. Straccia. A logic-based computational method for the automated induction of fuzzy ontology axioms. *Fundamenta Informaticae*, 124(4):503–519, 2013.

[29] A. Llaves, H. Michels, P. Maué, and M. Roth. Semantic event processing in ENVISION. In *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics*, page 25. ACM, 2012.

[30] J. R. Lloyd, D. Duvenaud, R. Grosse, J. B. Tenenbaum, and Z. Ghahramani. Automatic construction and natural-language description of nonparametric regression models. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1242–1250. AAAI Press, 2014.

[31] S. Lohmann, S. Negru, F. Haag, and T. Ertl. Visualizing ontologies with VOWL. *Semantic Web*, 7(4):399–419, 2016.

[32] C. Marinica and F. Guillet. Knowledge-based interactive postmining of association rules using ontologies. *Knowledge and Data Engineering, IEEE Transactions on*, 22(6):784–797, 2010.

[33] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. Patil, and D. Barton. Big data. *The management revolution. Harvard Bus Rev*, 90(10):61–67, 2012.

[34] A. Moraru and D. Mladenić. A framework for semantic enrichment of sensor data. *Journal of computing and information technology*, 20(3):167–173, 2012.

[35] A. Moraru, M. Pesko, M. Porcius, C. Fortuna, and D. Mladenic. Using machine learning on sensor data. *CIT. Journal of Computing and Information Technology*, 18(4):341–347, 2010.

[36] M. H. M. Noor, Z. Salcic, I. Kevin, and K. Wang. Enhancing ontological reasoning with uncertainty handling for activity recognition. *Knowledge-Based Systems*, 114:47–60, 2016.

[37] C. Otte. Safe and interpretable machine learning: a methodological review. In *Computational Intelligence in Intelligent Data Analysis*, pages 111–122. Springer, 2013.

[38] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical Black-Box Attacks Against Machine Learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[39] S. Peroni, D. Shotton, and F. Vitali. Tools for the Automatic Generation of Ontology Documentation: A Task-Based Evaluation. *Int. J. Semant. Web Inf. Syst.*, 9(1):21–44, Jan. 2013.

[40] B. Pfahringer. Random model trees: an effective and scalable regression method. Technical report, The University of Waikato, 2010.

[41] D. Pfisterer, K. Romer, D. Bimschas, O. Kleine, R. Mietz, C. Truong, H. Hasemann, A. Kroller, M. Pagel, M. Hauswirth, et al. SPITFIRE: Toward a Semantic Web of Things. *Communications Magazine, IEEE*, 49(11):40–48, 2011.

[42] M. Pugelj and S. Džeroski. Predicting structured outputs k-nearest neighbours method. In *Discovery Science*, pages 262–276. Springer, 2011.

[43] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[44] A. Rettinger, U. Lösch, V. Tresp, C. d'Amato, and N. Fanizzi. Mining the Semantic Web. *Data Mining and Knowledge Discovery*, 24(3):613–662, 2012.

[45] P. Ristoski and H. Paulheim. Semantic Web in data mining and knowledge discovery: A comprehensive survey. *Web semantics: science, services and agents on the World Wide Web*, 36:1–22, 2016.

[46] M. Ruta, F. Scioscia, and E. Di Sciascio. Enabling the Semantic Web of Things: framework and architecture. In *Sixth IEEE International Conference on Semantic Computing (ICSC 2012)*, pages 345–347. IEEE, IEEE, sep 2012.

[47] M. Ruta, F. Scioscia, F. Gramegna, G. Loseto, and E. Di Sciascio. Knowledge-based Real-Time Car Monitoring and Driving Assistance. In L. T. Nicola Ferro, editor, *20th Italian Symposium on Advanced Databases Systems (SEBD 2012)*, pages 289–294. Edizioni Libreria Progetto, jun 2012.

[48] M. Ruta, F. Scioscia, A. Pinto, E. Di Sciascio, F. Gramegna, S. Ieva, and G. Loseto. Resource annotation, dissemination and discovery in the Semantic Web of Things: a CoAP-based framework. In *Internet of Things (iThings/CPSCom), IEEE International Conference on*, pages 527–534. IEEE, 2013.

[49] F. Scioscia and M. Ruta. Building a Semantic Web of Things: issues and perspectives in information compression. In *Semantic Web Information Management (SWIM'09). In Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC 2009)*, pages 589–594. IEEE Computer Society, 2009.

[50] F. Scioscia, M. Ruta, G. Loseto, F. Gramegna, S. Ieva, A. Pinto, and E. Di Sciascio. A mobile matchmaker for the Ubiquitous Semantic Web. *International Journal on Semantic Web and Information Systems*, 10(4):77–100, 2014.

[51] L. Serafini, I. Donadello, and A. d'Avila Garcez. Learning and Reasoning in Logic Tensor Networks: Theory and Application to Semantic Image Interpretation. In *2017 Symposium on Applied Computing*, pages 1252–130. ACM, 2017.

[52] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[53] M. Stocker, M. Ronkko, and M. Kolehmainen. Situational knowledge representation for traffic observed by a pavement vibration sensor network. *Intelligent Transportation Systems, IEEE Transactions on*, 15(4):1441–1450, 2014.

[54] P.-Y. Vandenbussche, G. A. Atemezing, M. Poveda-Villalón, and B. Vatant. Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. *Semantic Web*, 8(3):437–452, 2017.

[55] J. Waliszko, W. T. Adrian, and A. Ligęza. *Traffic Danger Ontology for Citizen Safety Web System*, pages 165–173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[56] Y. Wang, Y. Tian, and K. Hu. Semantic Manipulations and Formal Ontology for Machine Learning based on Concept Algebra. *International Journal of Cognitive Informatics and Natural Intelligence*, 5(3):1–29, 2011.

[57] D. C. Wimalasuriya and D. Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 36(3):306–323, 2010.

[58] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[59] Z. Wu, Y. Xu, Y. Yang, C. Zhang, X. Zhu, and Y. Ji. Towards a Semantic Web of Things: A Hybrid Semantic Annotation, Extraction, and Reasoning Framework for Cyber-Physical System. *Sensors*, 17(2):403, 2017.

[60] N. Zhang, H. Chen, X. Chen, and J. Chen. Semantic framework of Internet of Things for smart cities: case studies. *Sensors*, 16(9):1501, 2016.