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. New Internet of Things (IoT) applications and services more and more rely on an intelligent understanding of the environment from 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 meaningful representations and explanations of taxonomies. This paper proposes a framework for a 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 by treating the typical classification problem of ML as a resource discovery. Outputs of classification are endowed with machine-understandable descriptions in standard Semantic Web languages, while explanation of matchmaking outcomes motivates confidence on results. A case study on road and traffic analysis 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 and capturing 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 exponentially grows-up. More than ever, effective methods are needed to treat data streams with the final goal to give a meaningful interpretation of retrieved information.

The *Big Data* label was coined to denote the research and development of data mining techniques and management infrastructures to deal with "volume, velocity, variety and veracity" issues [22] 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 with advances of pervasive technologies and Web-scale data management architectures, so that undeniable benefits have been produced from the data analysis point of view. Nevertheless, some not negligible weaknesses are still evident with respect to the increasing complexity and heterogeneity of pervasive computing challenges. Particularly, the lack of meaningful, machine-understandable characterization of outputs from classical ML techniques is a prominent limit for a possible exploitation in fully autonomic application scenarios.

This paper introduces an overall framework aiming to enhance classical ML analysis on IoT data streams, associating semantic descriptions to information retrieved from the physical world, as opposed to trivial classification labels. The basic idea is to treat a typical ML classification problem like an ontology-driven resource discovery. Steps include building a logic-based characterization of statistical data distributions and performing a fine-grained event detection, exploiting non-standard reasoning services for matchmaking [31].

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

The proposal grounds on both ideas and technologies of distributed knowledge-based systems [28], 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. Furthermore, the proposed theoretical model leverages an advanced matchmaking on metadata stored in sensing and capturing devices dipped in a context, lacking fixed knowledge bases. Inference tasks are distributed among devices which provide minimal computational capabilities. Stream reasoning techniques provide the groundwork to harness the flow of semantically annotated updates inferred from low-level data, in order to enable adaptive context-aware behaviors.

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. The approach was tested and validated in a case study for road and traffic monitoring on a real data set collected for experiments. Results were compared to classic ML algorithms in order to evaluate performance. The test campaign and early experiments preliminary assess both feasibility and sustainability of the proposed approach.

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

2. Motivation

Main motivation for this paper moves from the evidence of actual limits in the IoT. In spite of pervasiveness (miniaturization) and connectivity (interconnection capability) strengthen physical infrastructures, large data corpuses materialize without having really the possibility of analyze them in depth locally. Commonly adopted data mining techniques have two main drawbacks: i) they basically carry out no more than a classification task and ii) their precision is increased if applied on very big data amounts so making unfeasible an on-line analysis. These elements prevent *de facto* the possibility of realizing *thinking things*: the IoT is interpreted almost exclusively as sensing by the environment while is really veiled the possibility of making decisions and taking actions locally after the sensing stage.

It should be considered that in IoT scenarios, information is gathered through micro-devices attached to everyday items or deployed in given environments and interconnected wirelessly. Basically, due to their small size, such *objects* have minimum processing capabilities, a small storage and low-throughput communication capabilities. They continuously produce raw data whose volume makes necessary to be processed by advanced remote applications. An intelligent interpretation of retrieved information reduces dimensions and possibly generates decisions on what detected, even if at a remote stage. Classical Machine Learning techniques have been largely used for that, but their main weakness is in the lack of a telling and significant representation of revealed events.

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 ideas and technologies is bringing about the so-called Semantic Web of Things (SWoT) [28]. This paradigm aims to enable novel classes of intelligent applications and services grounded on Knowledge Representation (KR), exploiting semantic-based automatic inferences to infer implicit information starting from an explicit event and context detection [30].

By associating a structured and machine-understandable description in standard Semantic Web languages, each classification output could assume a non-ambiguous meaning. Furthermore, explanation capabilities provided by the underpinning semantic matchmaking allow to justify outcomes, so increasing confidence in system response. If pervasive micro-devices are capable of efficient on-board processing on the locally retrieved data, they can describe themselves and the context where they are located toward external devices and applications. This would enhance interoperability and flexibility, facilitating pervasive knowledge-based systems with high degrees of autonomicity not yet allowed by typical IoT infrastructures and procedures.

Two important consequences are induced. First of all also the 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 embedded micro-components, providing users with context-aware personalized task and decision support. Secondly, even if machine learning techniques, algorithms and tools have enabled novel classes of analyses (specially useful for Big Data Internet of Things perspective), the exploitation of logic-based and approximate discovery strategies –leveraging non-exact matching results– compensate possible faults in capturing activities, device volatility and the unreliability of wireless communications rolling out resilient IoT infrastructures really versatile for a widespread application.

3. Background

This section briefly recalls notions on Machine Learning and Description Logics, in order to make the paper self-contained and easily understandable. Then it discusses relevant related work.

3.1. Basics of Machine Learning

Machine Learning (ML) [36] is a branch of Artificial Intelligence which aims to build systems capable of learning from past experience. Differently from *expert systems*, ML algorithms and approaches are usually data-driven, inductive and general in nature; they are defined and applied to make predictions or decisions in some general class of tasks, *e.g.*, spam filtering, handwriting recognition or activity detection.

Three major categories of ML problems exist:

- *Classification*[1], consisting in the association of an observation (sample) to one of a set of possible categories (classes), *e.g.*, an e-mail message is spam or not. Classification has therefore a discrete n-ary output. This is the main problem considered in this paper.
- *Regression*, defined as the estimation of the relationship of a dependent variable from one or more independent variables, *e.g.*, predicting the purchase cost of a house considering its size, age, location and other features. In general, both the inputs and the output in regression can vary in continuous value ranges.
- *Clustering, i.e.*, dividing a set of observations into groups (clusters), which maximize the similitude of samples within each group and the difference between groups. Many pattern recognition problems are based on clustering.

The implementation of a ML system typically includes two main stages: *training* and *testing*. In the training phase, the adopted ML algorithm builds a *model* of the particular problem inductively from *training data*. Based on the model, the algorithm then produces outputs most likely matching the current inputs in the testing phase, which is executed during system validation in order to check fitness of the learning system. In particular, evaluation of classification performance is based on considering one of the output class 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.

With the above definitions, the following performance metrics for 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 $A = \frac{TP+TN}{TP+FP+TN+FN}$

Each available dataset to be classified is divided into a *training set* for model building and a *test set* for validation. There exist several approaches for selecting properly training and test components. Among others, in *k-fold cross-validation*, the dataset is partitioned into 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 and a smaller one to the test set.

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

ML methods can be divided into supervised or unsupervised. *Supervised* techniques require a relatively large corpus of labeled data to be built for training, usually by hand. Furthermore, the resulting models achieve good accuracy only for the specific scenarios they are built for. They are not reusable and scalable when conditions change. Conversely, *unsupervised* methods try to build models directly from unlabeled data, and are prevalent in clustering problems. The availability of large data collections associated with partial human annotation has recently turned the attention to *semi-supervised* learning [37]. By combining small-scale expert labeled data and large-scale unlabeled data based on certain assumptions, semi-supervised methods try to find the best tradeoff between system accuracy and required human and computational effort.

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

- *C4.5 decision tree* [26]: it adopts a greedy top-down approach for building the classification tree, starting with the creation of the root node. At each node, the information gain for each attribute is calculated and the attribute with the highest information gain is selected. The gain is defined as reduction in entropy caused by splitting the instances based on values taken by the attribute.
- *Functional Tree* [11,16]: a classification tree with logistic regression functions at the inner nodes and leaves. The algorithm can deal with binary and multiclass target variables, numeric and nominal attributes and missing values.
- *K-Nearest Neighbors, KNN* [1], an instance-based learning algorithm. It locates the k nearest instances to the input instance 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 works well with both nominal and numerical features.
- *Random Tree* [25]: it combines two other ML algorithms, 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 are an *ensemble learning* approach which builds several decision trees and picks the mode of their outputs.

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 First Order Logic [2]. Basic DL syntax elements are:

- *concept* (*a.k.a. class*) names, standing for sets of objects, *e.g.*, *vehicle, road, acceleration*;
- *role* (*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* is a pair $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, consisting 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 $\Delta \times \Delta$, and every individual to an element of Δ. We assume different individuals are mapped to different elements of Δ, *i.e.*, if $a \neq b$ then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ (*Unique Name Assumption*, UNA).

These elements can be combined using *constructors* to form concept and role *expressions*. Each DL has a different set of constructors. A constructor used in every DL is the *conjunction* of concepts, usually denoted as $\sqcap$; some DLs include also *disjunction* $\sqcup$ and *complement* $\neg$. Roles can be combined with concepts using *existential role quantification* (*e.g.*, $car \sqcap \exists hasEngine.DieselEngine$, which indicates the set of cars with a Diesel engine) and *universal role quantification* (*e.g.*, $vehicle \forall hasPneumatic.SnowTire$, which describes vehicles equipped only with snow tires). Other constructs may involve counting, as *number restrictions*: $car \sqcap \leq 2\, hasSeat$ denotes cars having at most 2 seats, and $vehicle \sqcap \geq 7\, hasSeat$ describes vehicles with at least 7 seats. 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 individual axioms (*a.k.a.* facts) forms an *Assertion Box (TBox)*, which together with its reference TBox constitutes a *Knowledge Base, KB*.

Adding new constructors makes DL languages more expressive. Nevertheless, this usually leads to a growth in computational complexity of inference services [4]. Hence a tradeoff is needed. 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 nonstandard inference services. Syntax and semantics of $\mathcal{ALN}$ constructors are reported in Table 1, along with corresponding elements in the RDF/XML serialization of the Web Ontology Language (OWL 2)[2] standard for the Semantic Web. OWL also supports *annotation properties* associated to class and property names, *e.g.*, for comments and versioning information.

3.3. Related work

In IoT scenarios, smart interconnected objects gather data samples, which can be used to identify and predict many real-world phenomena exhibiting patterns. Extracting high-level information from the raw data captured by sensors translating in machine-understandable languages has several interesting applications. The work [12] surveyed requirements, solutions and challenges in the area of information abstraction and presents an efficient workflow based on the current state of the art.

Semantic Web research addressed the task of describing sensor and data features through ontologies. *SSN-XG* [7] is the most relevant and widely accepted proposal. It is general enough to adapt to different applications and is compatible with the Open Geospatial Consortium (OGC) Sensor Web Enablement (SWE) standards at the sensor and observation levels [3]. OGC SWE was used in several frameworks aimed at granting access to sensor data as RESTful services or Linked Data [14,9]. The problem of semantic data flow compression in limited resource spaces was faced in [10] by developing a scalable middleware platform to publish semantically-annotated data streams on the Web through HTTP.

Unfortunately, the above solutions only allowed elementary queries in SPARQL fragments on RDF annotations. More effective techniques such as ontology-based *Complex Event Processing* (CEP) [5] exploited a shared domain conceptualization to define events and actions to be run on an event processing engine. Also the *ENVISION* project [19] combined CEP with semantic technologies to perform Semantic Event Processing from different sources: reasoning was used to process the incoming facts which populated a knowledge base. Using KR techniques on large amounts of instances could be in fact useful to annotate raw data and produce high-level descriptions in a KB suitable for advanced reasoning, aiming to improve standard data mining and ML algorithms [27]. In [21] a post-processing of ML operations based on ontology consistency check aimed to improve results of association rule mining. Semantically inconsistent associations were pruned and filtered out leveraging on logic reasoning for that. Extensions to standard reasoning algorithms, supporting uncertainty and time relationships, have been also proved as effective in tasks such as activity recognition [23].

Some of the most successful ML methods, such as Artificial Neural Networks (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 voluntary input manipulation [15,24]. Research efforts to build decipherable results of ML techniques and systems are therefore growing. A conceptually simple approach is to exploit ensemble learning combining multiple low-dimensional submodels, where each individual submodel is simple enough to be verifiable by domain experts [24]. In [17] Bayesian learning was used to generate lists of rules in the *if* ... *then* form, which can provide readable reasons for their predictions. The method was found to be competitive with state-of-the-art techniques in a stroke prediction task over a large dataset, although training time appears as rather long for IoT scenarios. The regression tool in [20] is able to translate automatically components of the model into 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 parameters) in each regression model, thus allowing the selection of the desired tradeoff between accuracy and ease of interpretation.

[2] 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>`	$A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

Semantic-enhanced ML methods can achieve the same goals through formal logic-based descriptions of models in order to develop explanatory functionalities, so increasing users' trust. Furthermore, they can be integrated into larger cognitive systems, where models and predictions are used for automated reasoning. Particularly, *Semantic data mining* refers to data mining tasks which systematically incorporate domain knowledge into the process. The survey [8] included 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 have been successfully used in all steps of a typical data mining workflow. In Ontology-Based Information Extraction (OBIE) [35] they are also exploited to annotate the output of data mining. In [33], the authors proposed an approach using wireless sensor networks and ontologies to represent and infer knowledge about traffic conditions. Raw data were classified through an ANN and mapped to ontology classes for performing rule-based reasoning. In [6], an unsupervised model was used for classifying Web Service datatypes in a large number of ontology classes, by adopting an extended ANN. Also in this case, however, mining was exploited only to map data to a single class. Promising semantic-based approaches also include fuzzy DL learning [18], concept algebra [34] and tensor networks based on Real Logic [32]. While their prediction performance appears good, computational efficiency must still be evaluated completely before considering them suitable for IoT scenarios.

4. Automatic identification of events via knowledge representation

The proposed framework 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-based enhancements grounded on DLs change the way each step is performed.

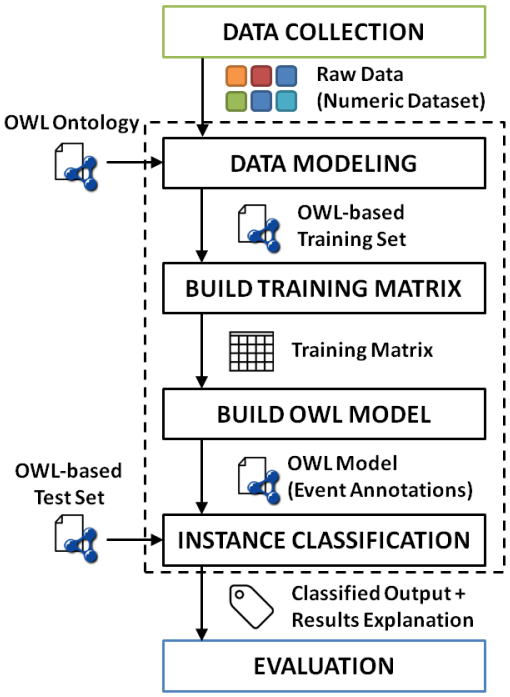


Fig. 1. Framework architecture

Individual steps of the devised methodology are outlined hereafter.

4.1. Ontology and data modeling

The workflow starts with raw data gathered *e.g.*, by sensors dipped in a given environment for extracting several different parameters, generally named *features*. In order to support semantic-based data annotation and interpretation, an ontology $\mathcal{T}$ models the domain conceptualization along properly specified *patterns*. $\mathcal{T}$ is assumed acyclic and expressed in the moderately expressive $\mathcal{ALN}$ DL. This is required by the subsequent non-standard inferences for semantic matchmaking [31]. For each measuring parameter, $\mathcal{T}$ must include a concept hierarchy (each one with its own properties), as in Figure 2, forming a partonomy of the topmost concept. In other words, each parameter will be represented via a class/subclass taxonomy featuring all significant value ranges and configurations it can have in the domain of interest. The depth of the hierarchy and the breadth of each level will be chosen by the knowledge modeler; they will typically be 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 were investigated to represent data ranges for the measured parameters within the knowledge base[3]. In the first one, each data range associated to a concept was 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 `Level_2_RPM` –corresponding to values from 801 to 900 engine revolutions per minute– was annotated as in Figure 3(a). Afterwards, this approach was modified: the modeling preserves the hierarchy of concepts, but to the range of potential variability of each measured parameter is given an explicit semantics by means of *number restrictions* associated to each subconcept. See the `Level_2_RPM` concept expressed in this way in Figure 3(b).

The second 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 [31]. Performance differences related to the above modeling approaches are described in Section 6.

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

4.2. Training

Like in classical ML, the goal of this step is to use training data to define the *model* to be used afterward by the ML algorithm to make predictions on test data. In the proposed approach, the model consists of a semantic annotation for each possible output class, connoting the observed event/phenomenon according to input data. 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 $j = 1\ldots,m$, then $\forall R.E_k$ is a concept component of C, for each E_k concept component of E.*

Every complex description is built by joining the concepts modeled in the previous step in conjunctive expressions.

The training phase works 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 feature 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 in the ontology.

Samples are processed sequentially by Algorithm 1 in order to build the so-called *Training Matrix* $\mathcal{M}$ (the

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

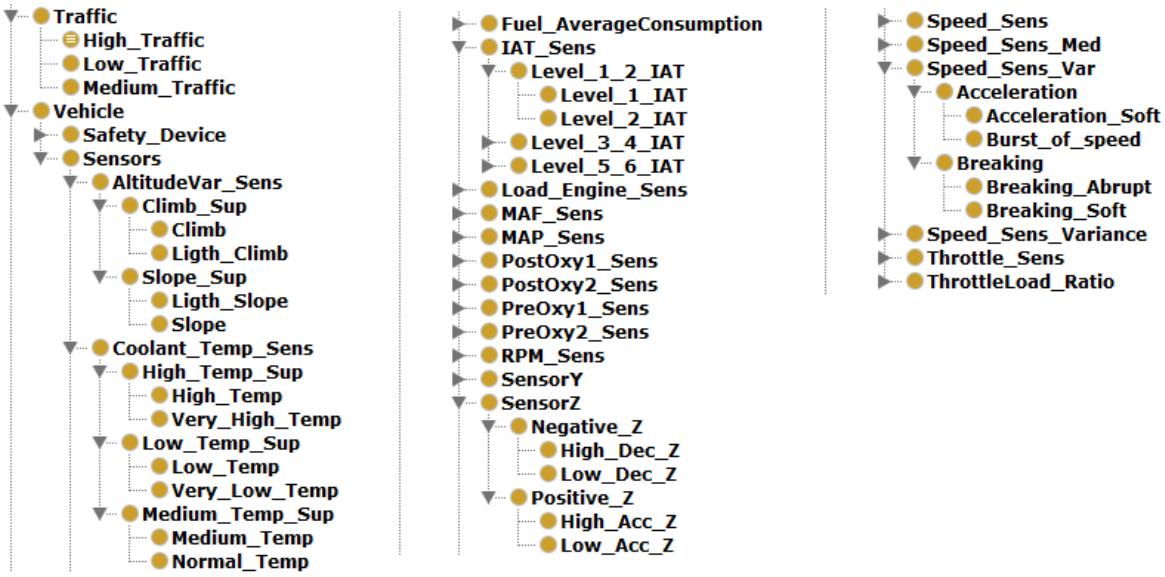


Fig. 2. Class hierarchy in the domain ontology for the case study

pseudocode uses a MATLAB-like notation for matrix access). $\mathcal{M}$ is a $(w+1) \times (k+1)$ matrix having all the different outputs on the first column, all the k distinct concept components occurring in the training set on the first row and, in each element, 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 was associated to that class), it appends a row to $\mathcal{M}$ initializing its values to zeros. Subsequently, for each concept components $C_{i,j}$, if $C_{i,j}$ is not yet in $\mathcal{M}$ (*i.e.*, no previous sample included that concept component), it appends a column and initializes 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).

$\mathcal{M}$ gives a complete picture of the training set. Each output class can now be defined as the 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 significant 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:

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

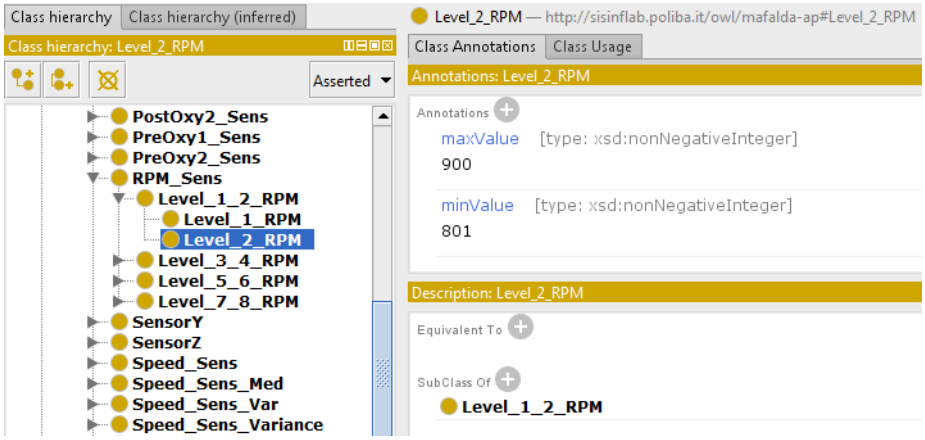
with $0 < \theta_{(i,j)} \leq 1 \;\; \forall i, j$ being adaptive ratios computed through *e.g.*, a cross-validation process on the training dataset.

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 was calculated in such a way, so as not to penalize 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 according: (i) 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" is much less frequent than "even road", normalization will increase all the values on the "uneven road" row in $\mathcal{M}$).

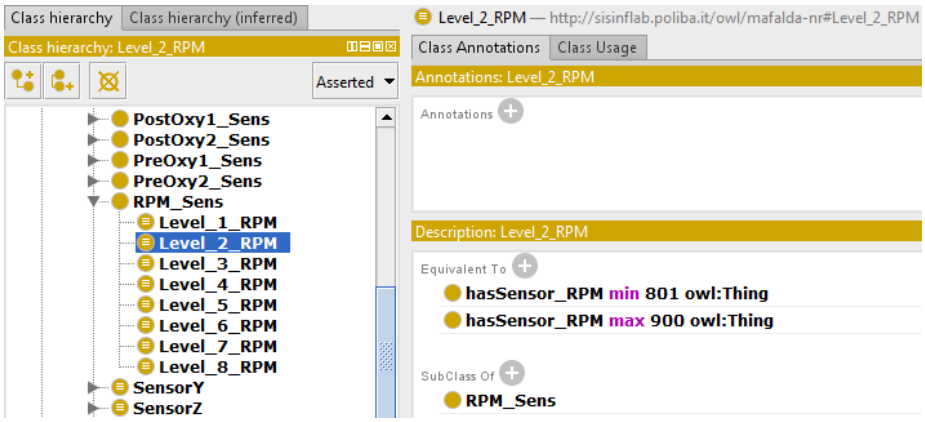
The adopted formula is:

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

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 concept



(a) Annotation properties



(b) Number restrictions

Fig. 3. Data range modeling

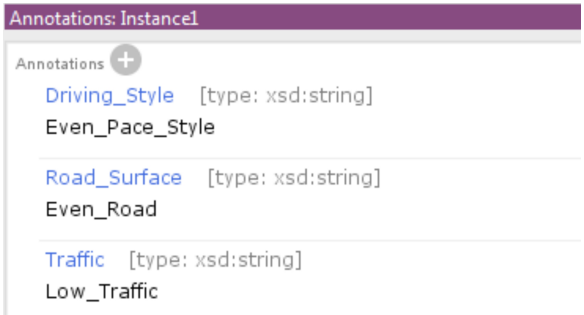


Fig. 4. Example of output class annotation

expression composed by the concepts occurring with a normalized frequency above the threshold.

Therefore this training approach produces a knowledge base with conceptual knowledge (the TBox) modeled by human experts and factual knowledge (the ABox) created automatically from the available data stream, with instances representing the events of interest the system should be 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 [31].

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.

Algorithm 1 Creation of the Training Matrix

Require:
– Description Logic $\mathcal{L}$;
– acyclic TBox $\mathcal{T}$;
– w output classes $O_1, O_2, \ldots, O_w$;
– training set $S = \{S_1, S_2, \ldots S_n\}$, with $S_i = (C_{i,1}, \ldots, C_{i,m}, O_i)\ \forall\ i = 1, \ldots, n$;
– all $C_{i,j}$ and O_i are expressed in $\mathcal{L}$ and satisfiable in $\mathcal{T}$.
Ensure:
– $\mathcal{M} : (w+1) \times (k+1)$ matrix of occurrences of the concepts for each observed output, where k is the total number of distinct concepts appearing in S

$\mathcal{M} := 0$ // 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 $\mathcal{M}$
    $r := r + 1$
    $u_r := r$
    $\mathcal{M}(u_r, 1) := O_i$
    initialize $\mathcal{M}(u_r, 2 : c)$ to zeros
  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 $\mathcal{M}$

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 considered as extensions respectively to *Satisfiability* and *Subsumption* standard inference services, which can only provide "yes/no" answers in KR systems.

The proposed framework first labels data of the instance 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 were defined. In case of *annotation properties*, the penalty score is computed via the following formula:

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

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, when using *number restrictions*, the function is defined as:

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

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.

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 opaque predictions. Furthermore, notice that the approach does not take the instance annotation directly as the output, because the inherent data volatility in IoT contexts could lead to inconsistent assertions, which would be impossible to reason on.

4.4. Evaluation

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 (such as accuracy, precision and recall), the graded nature of predictions in the proposed approach, *e.g.*, the average semantic distance of the predicted class from

the actual one, allows to evaluate applying typical error measures of regression analysis like 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. As a case study, a prototypical system for road and traffic monitoring was 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).

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[4] (OBD-II) port;
2. integrate environmental information;
3. detect potential risk factors.

Besides providing warnings, the detected knowledge allows giving suggestions to the driver and evaluating car efficiency and environmental impact in real time [29].

The Java language was chosen for the implementation, in order to be compatible with both Java SE (Standard Edition) and Android platforms. The prototype included the *Mini-ME* lightweight matchmaker [31], which provides the required inferences for the $\mathcal{ALN}$ DL (under the assumption of acyclic TBoxes). The above ML framework was used to extract high-level indications starting from a large number of low-level parameters acquired by the car via OBD-II and through the micro-devices embedded in the user smartphone, with the goal of accurately characterizing the overall system composed by driver, vehicle and environment.

A dedicated dataset was collected for further experiments[5]. Raw data were retrieved and stored using the *Torque Lite (OBD-II & Car)*[6] Android application on seven different routes: suburban, urban and mixed ones, with medium and long distance. An average of five traces per route were recorded, sampling OBD-II parameters and smartphone data at 1 Hz frequency. About 10,000 records were 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) were used.

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

- *Even*, *Slightly Uneven* or *Uneven* Road;
- *Low*, *Medium* or *High* Traffic;
- *Aggressive* or *Even Pace* driving style.

During the dataset creation, each driver who collected a trace was 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) were added to each record.

Analyzed data consisted 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 a percentage;
- *engine coolant temperatures*;

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

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

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

- *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 were represented in the domain ontology and divided in subclasses, each characterized by a value range. At the end of the training phase, the ABox was created automatically from the available data stream, with 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 was also developed to validate the framework in real-time usage. It is an evolution of [29], 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;
- 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.

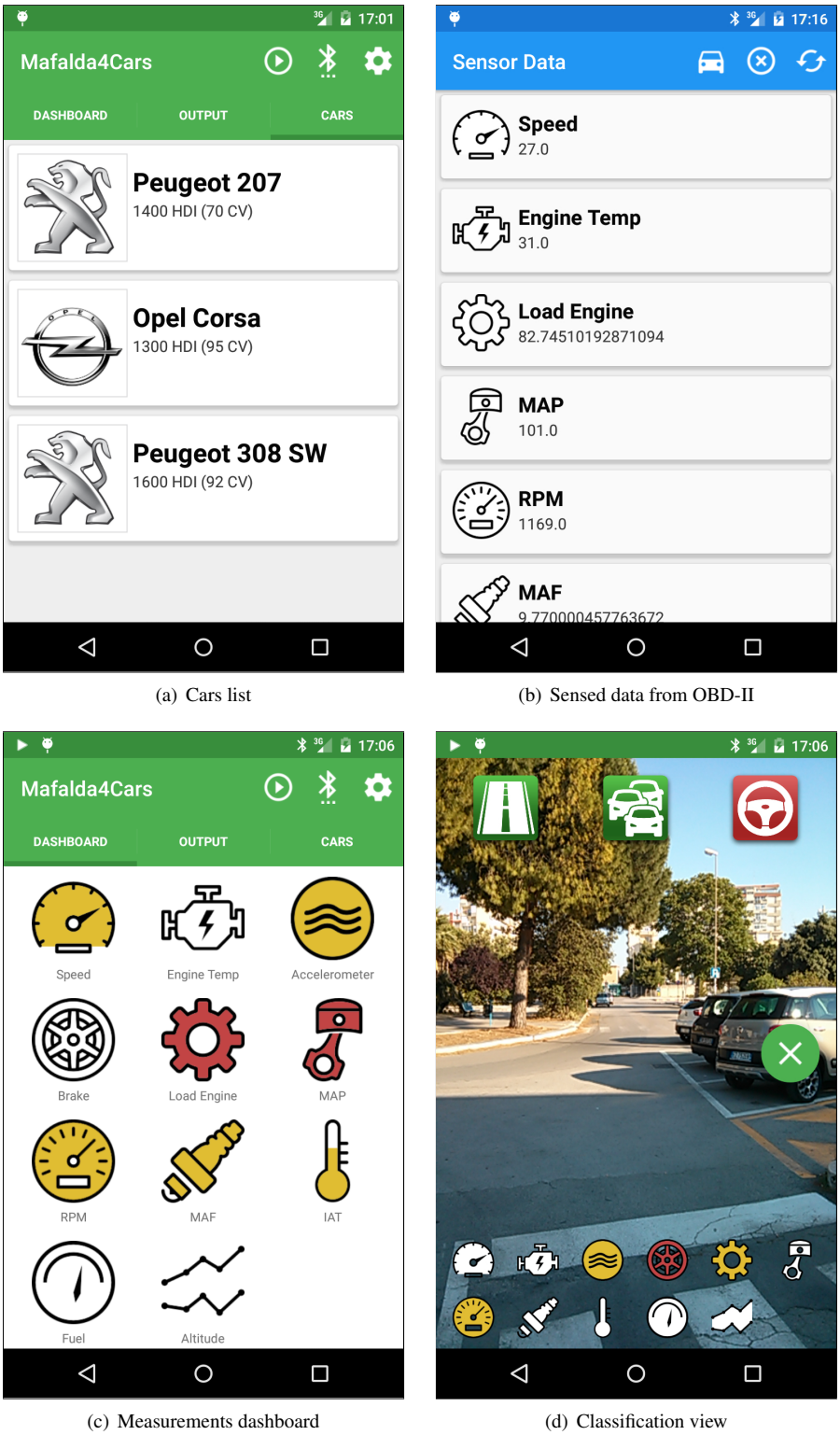
Moreover, the user can start the classification view in Figure 5(d). The smartphone camera viewfinder is used as background to allow the user to see the classification outputs without looking away from the road. The application queries vehicle information via OBD-II and executes the algorithm described in Section 4. The user interface shows at the bottom a compact device dashboard (like in Figure 5(c), but smaller) whereas at the top three large icons are displayed, 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 an even road and low traffic (green icons) and an aggressive driving style (red icon).

6. Experiments

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

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 was divided in a training set and a test set by holdout, mixing the records randomly in 70% and 30% ratios, respectively. The training set generated the model, while the test set allowed to evaluate the classification performance. Training and test set were 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 were 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* present lower values due to the disjunction of intervals in modeling concepts of ontology: semantic descriptions produced by the training phase were all similar, penalizing the later stages. Indeed, in the classification phase, the matchmaking between the output description generated from the training phase and the sample from test set tended to have increased penalty values due to disjoint *number restrictions*, frequently producing an incorrect classification output.

The same training and test sets were used with classical Machine Learning algorithms to compare and evaluate results obtained with the best configuration of the proposed approach (nicknamed *MAFALDA*, as *MAtchmaking Features for mAchine Learning Data Analisys*). The four algorithms recalled in Section 3.1



(a) Cars list

(b) Sensed data from OBD-II

(c) Measurements dashboard

(d) Classification view

Fig. 5. Mobile application screenshots

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

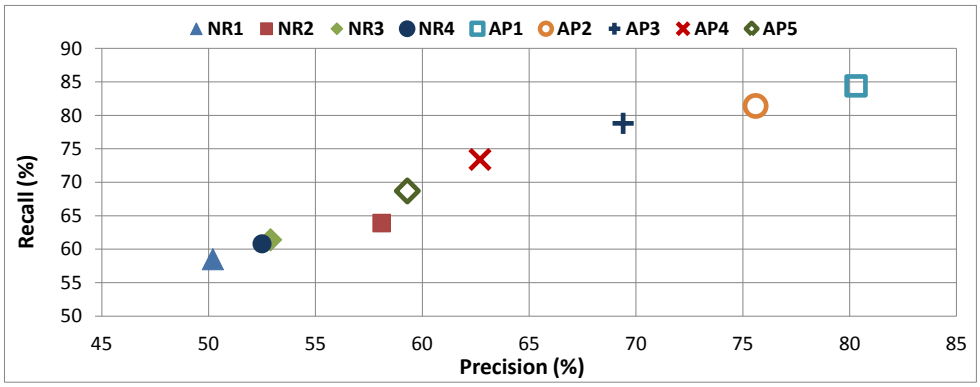


Fig. 6. Precision/recall plot

were used for the comparison, in the implementation of Weka[7] [13]:

- J48 implementation of C4.5;
- Functional Tree (FT);
- K-Nearest Neighbours (k-NN);
- Random Tree (RT).

Also in this case, each algorithm was used for testing different configurations obtained by setting conveniently parameters in Table 3. Results corresponding to the configurations with the highest accuracy are reported in Table 4. *MAFALDA* presented 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 representation.

Experimental analysis about processing time was carried out on three different platforms:

- 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 and 64-bit Java 8 SE Runtime Environment, build 1.8.0_31-b13;
- *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*[8], 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.

The test was executed using the first Peugeot 207 dataset consisting of 8615 records; 6030 used as training set and 2585 as test set. Each test was repeated five times and the average value was taken, as reported in Figure 7.

The overall process included several sub-steps:

[7] Weka version 3.6.12, http://www.cs.waikato.ac.nz/ml/weka/

[8] 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
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
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

Table 4
Comparison of ML algorithms

Algorithm	Precis.	Recall	F-Score	Accur.
J48	0.885	0.883	0.884	0.883
FT	0.884	0.880	0.882	0.876
k-NN	0.878	0.863	0.870	0.863
RT	0.879	0.879	0.879	0.879
MAFALDA	0.861	0.813	0.836	0.813

1. *Ontology Loading*: load and parse the OWL file containing the TBox $\mathcal{T}$;
2. *Data Mapping*: for each of the 6030 data records included within the training set, identify the concept subclass(es) corresponding to the parameter values;
3. *Matrix Creation*: create/update the Training Matrix using the concepts identified in the previous step;
4. *Matrix Normalization*: normalize the matrix values and calculate the reference thresholds;
5. *OWL Model Creation*: starting from the normalized matrix, create the semantic annotations describing each event;
6. *Classification*: classify each of the 2585 data records included within the test set.

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

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 annotation) was 40 ms on PC, 630 ms on smartphone and 1.48 s on Raspberry, which can be deemed as acceptable also for mobile and embedded system. It is useful to point out that model training is performed only once after training set selection. Furthermore, processing time clearly appears as 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 was executed in about 0.22 ms on PC, 8 ms on mobile and 29 ms on Raspberry. This is a significant outcome because it suggests that the proposed approach is very responsive even with multiple features.

7. Conclusion and future work

This paper introduced a novel approach for semantic-enhanced machine learning on heterogeneous data streams in the Internet of Things. Mapping raw data

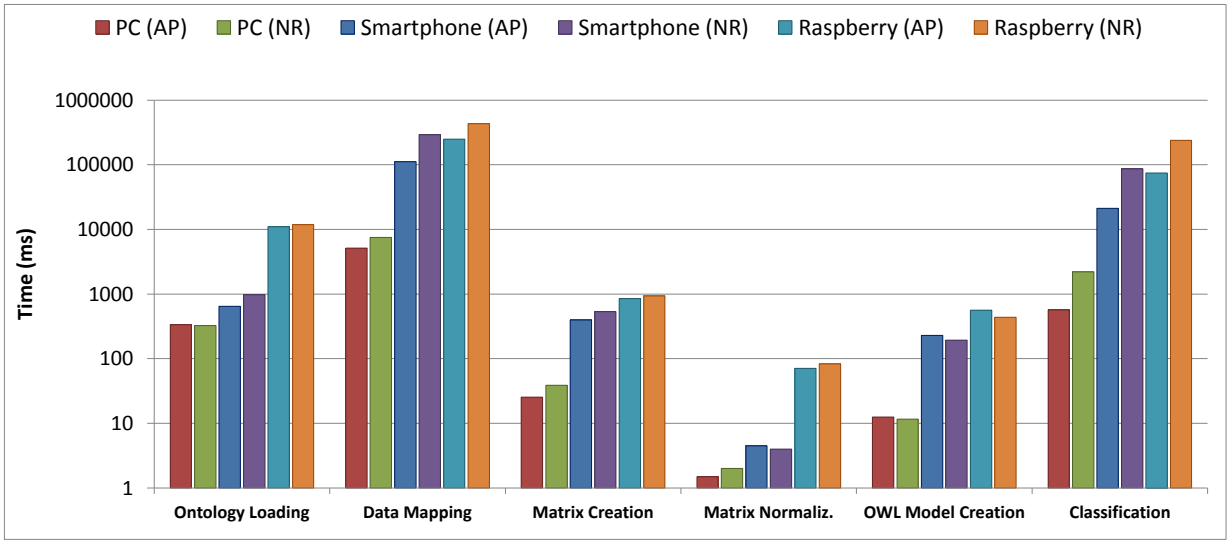


Fig. 7. Processing Time

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 matchmaking inferences enables a fine-grained event detection by treating the ML classification problem as a resource discovery.

A concrete case study on driving assistance was developed through data gathered from real vehicles via On-Board Diagnostics protocol (OBD-II) and exploiting sensing micro-devices (accelerometer, gyroscope, GPS) embedded on users' smartphones. A realistic dataset was so built for experimentation. Subsequent extensive evaluations allowed to assess the effectiveness of the proposed approach whose performance results were compared with state-of-the-art ML technologies, in order to highlight benefits and limits of the proposal.

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 will aim at a processing distributed on more than one node with a final merging step. This could reduce the communication overhead within a sensor network if intermediate nodes have storage capacity enough. Further variants could increase the flexibility of the proposed approach in the classification phase. For example it could be useful to investigate the possibility to create dynamically superclasses with a range that combine 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] D. Aha, D. Kibler, and M. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

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

[3] 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.

[4] 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.

[5] 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.

[6] E. S. Chifu and I. A. Letia. Unsupervised semantic annotation of Web service datatypes. In *Intelligent Computer Com-*

munication and Processing (ICCP), 2010 IEEE International Conference on, pages 43–50. IEEE, 2010.

[7] 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.

[8] 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.

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

[10] 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.

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

[12] 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.

[13] 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.

[14] 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.

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

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

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

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

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

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

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

[37] X. Zhu. Semi-supervised learning. In *Encyclopedia of Machine Learning*, pages 892–897. Springer, 2011.