Computers and Electrical Engineering 68 (2018) 1–13

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Ontology-based feature generation to improve accuracy of activity recognition in smart environments[☆]

A.G. Salguero[*,a], M. Espinilla[b]

[a] Universidad de Cádiz, Cádiz, Spain
[b] Universidad de Jaén, Jaén, Spain

ARTICLE INFO

Keywords:
Activity recognition
Smart environments
Ambient assisted living (AAL)
Activities of Daily Living (ADL)
Ontology
Data-Driven approaches
Knowledge-Driven approaches

ABSTRACT

In recent years, many techniques have been proposed for automatic recognition of Activities of Daily Living from smart home sensor data. However, classifiers usually use features created ad hoc. In this work, the use of ontologies is proposed for the fully automatic generation of these features. The process consists of converting the original dataset into an ontology and then combine all the concepts and relations in that ontology to obtain relevant class expressions. The high formalization of ontologies allows us to reduce the search space by discarding many meaningless expressions, such as contradictory or unsatisfiable expressions. The relevant class expressions are then used as features by the classifiers to build the classification model. To validate our proposal, we have used as reference the results obtained by four different classification algorithms that use the most commonly used features.

1. Introduction

A very important process at the core of smart environments is the sensor-based activity recognition [1–3]. This kind of activity recognition is based on recognizing the actions of one or more persons within an intelligent environment by using a flow of observed events that depend only on the current activity. Common activities of interest are Activities of Daily Living (ADLs) such as "bathing","sleeping" or "dinning". Objects or furniture can generate sensor events indicating, for example, the use of a faucet, the opening of a door, or the use of a light switch.

Approaches used for sensor-based activity recognition have been divided into two main kinds: Data-Driven (DDA) and Knowledge-Driven (KDA) approaches. DDA, are based on machine learning techniques in which a preexistent dataset of user behaviors is required. A training process is carried out, usually, to build an activity model which is followed by a testing process to evaluate the generalization of the model in classifying unseen activities [4]. The most remarkable features of the DDA are the capabilities of handling uncertainty and temporal information. DDA approaches need large annotated datasets for training and learning. In this context, it is interesting to mention the Open Data Initiative (ODI) [5] for Activity Recognition consortium that aims to create a structured approach to provide annotated datasets in an accessible format.

With KDA, an activity model is built through the incorporation of rich prior domain knowledge obtained from the application domain, using knowledge engineering and knowledge management techniques [6]. KDA has the advantages of being semantically clear, logically elegant, and easy to get started. In the context of KDA, ontologies for activity recognition have provided successful results [7]. In this kind of approach, interpretable activity models are built in order to match different object names with a term in an

[☆] Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. A. Sangaiah.
[*] Corresponding author.
E-mail addresses: alberto.salguero@uca.es (A.G. Salguero), mestevez@ujaen.es (M. Espinilla).

https://doi.org/10.1016/j.compeleceng.2018.03.048
Received 31 May 2017; Received in revised form 26 March 2018; Accepted 27 March 2018

ontology that is related to a particular activity.

Ontologies can be seen as structured vocabularies that explain the relations among their terms (or classes). They are formed by concepts and relations that can be combined to form more complex class expressions. Because of the high rigidity of the logic behind ontologies, some hybrid approaches have been developed [8,9] that take advantage of the main benefits provided by DDA and the use of ontologies. Thereby, ontological ADL models capture and encode rich domain knowledge and heuristics in a machine understandable and processable way.

A hybrid approach for activity recognition is presented in this paper, where ontologies are used to automatically generate the features for the ADL classifiers. The features correspond to class expressions that have been created by combining the concepts and relations in the ontology, according to a given set of rules. Contradictory or unsatisfiable class expressions can be detected and discarded, greatly reducing the feature search space. Other similar proposals can be found in literature [10–12], but this is the only proposal that performs the process in a fully automatic way. In addition, and unlike the rest, the proposal presented in this paper does not incorporate external data to the knowledge base. It is based solely on the data available in the original dataset. The proposal presented in this paper could be considered as an approach for the problem of feature learning. However, the goal of feature learning is often to reduce the dimensionality of the dataset, selecting or aggregating features in order to produce low-dimensional versions of the original datasets [13], whereas our proposal expands the set of existing features, looking for new relevant features that help us to find hidden patterns in the dataset.

To evaluate the quality and efficiency of the methodology proposed in this work an experiment has been carried out, in which the datasets proposed in [14–16] have been used. The results obtained by using the classic approach for the recognition of ADL have been used as reference to measure the performance of our proposal. In this approach, the features are handcrafted, and usually represent the state of the sensors during the activity. Therefore, each sensor provides a single feature to the algorithm that generates the classification model. There is often more relevant information in the dataset, such as the order in which the sensors change. However, this kind of information is not usually taken into account because it requires the development of *ad hoc* applications [17]. The proposal presented in this paper automatically discovers relevant sequences of sensor changes as it generates more and more class expressions.

The remainder of the paper is structured as follows: Section 2 reviews the binary sensor data within the smart environment used in this proposal with the simple transformation into feature vectors. Furthermore, notions about ontologies are revised to understand our proposal as well as related works. Section 3 proposes the methodology to extend the set of feature vectors by means of an ontology. Section 4 presents an empirical study that analyzes our proposed methodology of extended feature vector in terms of accuracy based on three popular datasets by using the ontology. In Section 5, the results obtained are analyzed and discussed. Finally, in Section 6, conclusions and future works are presented.

2. Background

In this section, firstly, the process to transform a sensor data stream generated by a smart environment into classical feature vectors used by DDA to recognize activities is reviewed. Furthermore, the three datasets used to evaluate our proposal are described. Then, some relevant concepts related to ontologies are reviewed in order to understand our proposed methodology to extend the feature vectors with the inferred knowledge by the ontology. Finally, related works are also presented at the end of this section.

2.1. From sensor data stream to feature vectors. Smart environment datasets

Usually, feature vectors generated by a smart environment are computed from the temporal sensor data stream that is discretized into a set of time windows, denoting each time window by W^k, which is limited by each activity. The set of activities are denoted by $A = \{a_1, ..., a_i, ..., a_{AN}\}$, being AN the number of activities of the dataset.

Each feature vector is denoted by F^k and has $N_S + 1$ components, N_S being the number of sensors in the dataset denoted by $S = \{s_1, ..., s_j, ..., s_{N_S}\}$. Therefore, each computed feature vector is defined by the following expression:

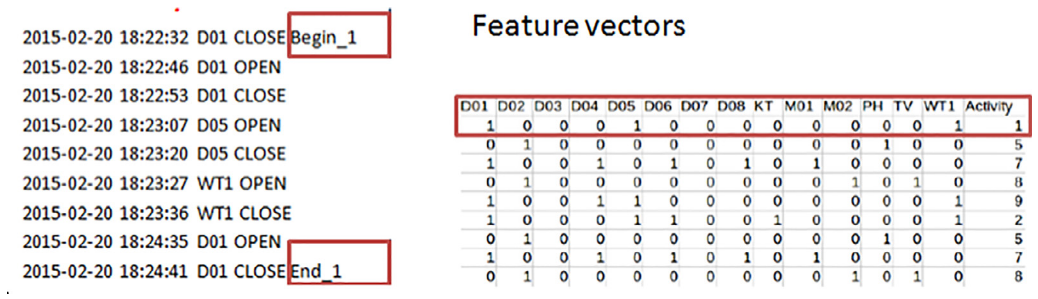
$$F^k = \left\{ f_1^k, ..., f_j^k, ..., f_{N_S}^k, f_{N_S+1}^k \right\}$$

being $f_j^k; j = \{1, ..., N_S\}$ a binary value that indicates if the sensor s_i was fired at least once, 1, or was not fired 0 in this time window W^k (see Fig. 1). The last component $f_{N_S+1}^k \in A$ indicates the activity carried out in the time window W^k.

In this paper, three popular activity recognition datasets of smart environments are used to evaluate our proposal, which are described below.

The first dataset was proposed in [14]. This dataset is composed by binary temporal data from a number of sensors, which monitored the ADLs carried out in a home setting by a single inhabitant. This dataset was collected in the house of a 26-year-old male who lived alone in a three-room apartment. This dataset contains 245 activities that are annotated in the stream of state-change sensors generated by 14 binary sensors. In this dataset, seven activities are classified: leave house, use toilet, take shower, go to bed, prepare breakfast, prepare dinner and, finally, getting a drink.

The second dataset was proposed in [15] that represents a sensor data stream in the Washington State University smart apartment. The data represents 20 participants performing eight ADL activities in the apartment. The activities were performed individually and sequentially. Each participant performed the same set of activities in any order. This dataset contains 178 activities that are annotated in the stream of state-change sensors generated by 45 sensors, three temperature sensor were omitted in the



```
2015-02-20 18:22:32 D01 CLOSE Begin_1
2015-02-20 18:22:46 D01 OPEN
2015-02-20 18:22:53 D01 CLOSE
2015-02-20 18:23:07 D05 OPEN
2015-02-20 18:23:20 D05 CLOSE
2015-02-20 18:23:27 WT1 OPEN
2015-02-20 18:23:36 WT1 CLOSE
2015-02-20 18:24:35 D01 OPEN
2015-02-20 18:24:41 D01 CLOSE End_1
```

Feature vectors

D01	D02	D03	D04	D05	D06	D07	D08	KT	M01	M02	PH	TV	WT1	Activity
1	0	0	0	1	0	0	0	0	0	0	0	0	1	1
0	1	0	0	0	0	0	0	0	0	0	1	0	0	5
1	0	0	1	0	1	0	1	0	1	0	0	0	0	7
0	1	0	0	0	0	0	0	0	0	1	0	1	0	8
1	0	0	1	1	0	0	0	0	0	0	0	0	1	9
1	0	0	0	1	1	0	0	1	0	0	0	0	1	2
0	1	0	0	0	0	0	0	0	0	0	1	0	0	5
1	0	0	1	0	1	0	1	0	1	0	0	0	0	7
0	1	0	0	0	0	0	0	0	0	1	0	1	0	8

Fig. 1. Partial sensor data stream of a dataset and its computed feature vector.

evaluation due to the fact that they do not provide information on the recognition of activities. In this dataset eight activities are classified: answer the phone, choose outfit, clean, fill medication dispenser, prepare birthday card, prepare soup, watch DVD and water plants.

The third dataset was proposed in [16] that is located in the UC Irvine Machine Learning Repository. The dataset represents two participants performing ten ADL activities in their own homes. The activities were performed individually and this dataset is composed of two instances of data, each one corresponding to a different user and completed in 35 days. In this dataset, the number of sensors are 12, although two of them are never fired in the case of the second participant. Ten activities are classified: breakfast, dinner, leaving, lunch, showering, sleeping, snacking, spare time TV and grooming.

The classical feature vectors obtained with these datasets will be expanded through ontologies to improve the accuracy of the results with DDA. In the next subsection, ontology basics are reviewed.

2.2. Ontologies

Ontologies are used to provide structured vocabularies that explain the relations among terms, allowing an unambiguous interpretation of their meaning. Ontologies are formed by concepts (or classes) which are, usually, organized in hierarchies, the ontologies being more complex than taxonomies because they not only consider *type-of* relations, but they also consider other relations, including *part-of* or domain-specific relations.

In an ontology, the symbol $\top$ stands for the *top* concept of the hierarchy, all other concepts being subsets of $\top$. The *subsumption* relation is usually expressed using the symbol $A \sqsubseteq B$, meaning that the concept A is a subset of the concept B. Concepts can also be specified as logical combinations of other concepts.

The semantic of operators for combining concepts is shown in Table 1, where C, C_1, $C_2 \sqsubseteq \top$, R is a relation among concepts, Δ_I is the domain of individuals in the model and I is an interpretation function.

The main advantage of ontologies is that they codify knowledge and make it reusable by people, databases, and applications that need to share information. Due to this, the construction, the integration and the evolution of ontologies have been critical for the Semantic Web. However, obtaining a high quality ontology largely depends on the availability of well-defined semantics and powerful reasoning tools.

Regarding Semantic Web, a formal language is OWL, which is developed by the World Wide Web Consortium (W3C). Originally, OWL was designed to represent information about categories of objects and how they are related. OWL inherits characteristics from several representation languages families, including the Description Logics (DL) and Frames basically. OWL is built on top of the Resource Description Framework (RDF) and RDF Schema (RDFS). RDF is a data-model for describing resources and relations between them. RDFS describes how to use RDF to describe application and domain specific vocabularies. It extends the definition for some of the elements of RDF to allow the typing of properties (domain and range) and the creation of subconcepts and subproperties. The major extension over RDFS is that OWL has the ability to impose restrictions on properties for certain classes.

One of the main advantages of the high formalization of OWL is the possibility of using automated reasoning techniques. In 2009, the W3C proposed the OWL 2 recommendation in order to solve some usability problems detected in the previous version, keeping

Table 1
Semantic of OWL logical operators.

	DL syntax	Manchester syntax	Semantics
$\mathscr{I}$	$C_1 \sqcap C_2$	C_1*and* C_2	$(C_1 \sqcap C_2)^I = (C_1^I \cap C_2^I)$
$\mathscr{U}$	$C_1 \sqcup C_2$	C_1*or* C_2	$(C_1 \cup C_2)^I = (C_1^I \cup C_2^I)$
$\mathscr{C}$	$\neg C$	*not C*	$(\neg C)^I = \Delta_I \setminus C^I$
$\mathscr{S}$	$\exists R.C$	*R some C*	$(\exists R.C)^I = \{x \mid \exists y. \langle x, y \rangle \in R^I \wedge y \in C^I\}$
$\mathscr{A}$	$\forall R.C$	*R only C*	$(\forall R.C)^I = \{x \mid \forall y. \langle x, y \rangle \in R^I \rightarrow y \in C^I\}$
$\mathscr{X}$	$\leq nR.C$	*R maxn C*	$(\geq nR.C)^I = \{x \mid card\{y. \langle x, y \rangle \in R^I \wedge y \in C^I\} \leq n\}$
$\mathscr{M}$	$\geq nR.C$	*R minn C*	$(\leq nR.C)^I = \{x \mid card\{y. \langle x, y \rangle \in R^I \wedge y \in C^I\} \geq n\}$

the base of OWL. So, OWL 2 adds several new features to OWL, some of the new features are syntactic sugar (e.g., disjoint union of classes) while others offer new expressivity, including: increased expressive power for properties, simple metamodeling capabilities, extended support for datatypes, extended annotation capabilities, and other innovations and minor features.

2.3. Related works

We can find in the literature many ontologies for the description of ADLs. However, all of them are designed to be as expressive as possible, defining a huge amount of classes and properties. This makes more difficult for our methodology to find relevant class expressions, since the search space grows exponentially. The ontology proposed in [18], for example, contains one hundred and sixty-five predefined activities and thirty-three different types of predefined events ("atomic activities"). It also incorporates other types of entities that are specific for the dataset, such as the person performing the activity and his or her posture. Also, all properties are flat, i.e., without any characteristic (inverse, functional...), because the reasoning about the order of events is done in an external rules system. The same applies to the ontology proposed in [6], where there are properties to associate a sensor with the object in which it is located and even its manufacturer. Their authors distinguish between simple and compound activities and they are organized in hierarchies. There are also dozens of activities and properties already predefined in the ontology that do not correspond to the activities in the datasets of the experiment described in Section 4. The ontology proposed in [19] is oriented towards the development of an ADLs monitoring system which can interact with the users through mobile networks. It includes concepts and properties to implement a message service between the user and the monitoring system. The ontology proposed in [20] is the most similar to the one used in this paper. However, it defines concepts to indicate the type of sensor or its location, which are not used in the experiment proposed in this work. Nonetheless, the biggest problem is the distinction made between basic and composite activities, because the class expressions become more complex and more of them have to be generated in order to find relevant features. Besides, there is no transitive property that allows to know all the events produced before or after a certain one. All the intermediate events have to be explicitly given in the class expression to relate two non-consecutive events, increasing the complexity of such expressions.

In the context of feature generation, a framework that generates new features for a movie recommendation dataset is proposed in [10]. They use that framework to construct semantic features from YAGO, a general purpose knowledge base that was automatically constructed from Wikipedia, WordNet and other semi-structured Web sources. Then, they manually define a set of static queries in SPARQL language that are used to add information to the original dataset, such as its budget, release date, cast, genres, box-office information, etc. Despite being a proposal similar to ours, it is important to note that the set of features are generated in a fully automatic process, thanks to the high formalization of ontologies, without the need for human interaction.

The authors in [11] also propose the use of ontologies to generate new features. They expand features from the original feature in a breadth first search manner considering some rules for semantically correct paths. Only concepts on outgoing paths from the original entity conforming to these patterns are considered as possible features in the further process. Although they plan to test their proposal with two ontologies, they are actually dealing with the underlying RDF graph of those ontologies. They do not make use of the inference mechanisms of ontologies nor the formal logic behind them. They just use the user defined relationships between concepts in the RDF graph in order to relate the original concept with the concepts in its context.

Paulheim [12] also proposes another technique that employs user defined relations between concepts in the RDF graph of ontologies for the automatic generation of features. Its main goal is the generation of possible interpretations for statistics using Linked Open Data. The prototype implementation can import arbitrary statistics files, and uses DBpedia for generating attributes in a fully automatic fashion. Furthermore, the author argues that their approach works with any arbitrary SPARQL endpoint providing Linked Open Data. The use of the inference mechanisms of ontologies is also very limited in this work.

Another key difference of our proposal with respect to those in [10–12] is that they propose the use of external knowledge to generate new features, whereas we only consider the information in the original dataset to do so.

The methodology proposed in this paper also shares some characteristics with Class Expression Learning (CEL) techniques. The algorithms for CEL are mainly used in the field of ontology engineering [21]. They can be used to suggest new class descriptions that are relevant to the problem while the ontologies are being developed. The objective of CEL algorithms is to determine new class descriptions for concepts that may be used to classify individuals in an ontology according to some criterion. More formally, given a class C, the goal of CEL algorithms is to determine a class description A such that $A \equiv C$. Given a set of positive and negative examples of individuals in an ontology, the learning problem consists on finding a new class expression or concept such that most of the positive examples are instances of that concept, whereas the negatives instances are not.

The main difference with respect to our proposal is that the result of CEL algorithms is always a DL class expression whereas the result of the proposed methodology is a set of DL class expressions, which do not always describe positive instances. Sometimes, the features of negative instances provide valuable information to the classification model. In our case, the entire set of generated class expressions are treated as features. The classifier may combine the DL class expressions as necessary, without the need for producing the result in form of logical axioms that describe positive instances. This is the reason why the classifiers based on our proposal perform better than the CEL algorithms. In addition, the rigidity of DL makes the classification model less tolerant to conflicting and incoherent situations due to faulty sensing hardware or communication problems [22].

3. Methodology

This section describes the proposed methodology. The purpose of our methodology is to extend the feature vectors processed by a smart environment to enrich these vectors through asserted and inferred knowledge in the ontology, improving the accuracy of

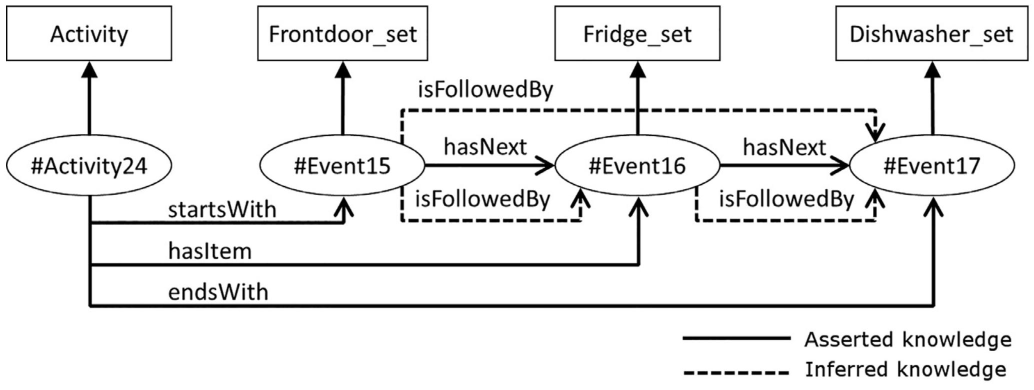


Fig. 2. Ontology example.

classifiers based on DDAs used in the recognition of activities against the unextended feature vectors.

3.1. An ontology for the description of ADL

This section presents the ontology developed for the description of ADL. The aim of this ontology is to provide a basic set of primitives that allow the representation of the information present in most datasets available in the literature. The set of primitives should be comprehensive enough to be able to represent all the activities in such datasets but should also be as brief as possible to facilitate the use of reasoners.

The ontology defines two basic disjoint concepts, *Activity* and *Event*, which respectively represent all the activities in the datasets and the activation of the sensors during these activities. In fact, the *Event* concept represents any situation reported by a sensor. It can also be used to represent the deactivation of a sensor or the report of its value at a certain instant. Each of the sensors in the dataset requires the creation of at least one subconcept of the *Event* concept to represent the events produced by that sensor. The application developed to convert datasets into ontologies creates two subconcepts for binary sensors. One of them represents the activation of the sensor and the other, which is optional, its deactivation. In the first case the suffix "_set" is appended to the end of the name of the concept, while the suffix "_clear" is appended in the second. When it comes to a numeric sensor the name of the concept is just the name of the sensor. The class $Frontdoorset \sqsubseteq Event$, for example, represents the set of events corresponding to the activation of the sensor on the front door for the dataset in [14].

Our proposal for representing activities is based on a list structure (see Fig. 2). However, the underlying RDF collections are unavailable because they are used in the RDF serialization of OWL. Although *rdf:Seq* is not illegal, it depends on lexical ordering and has no logical semantics accessible to a DL classifier [23].

In the literature, we can find some proposals for the representation of lists in OWL. In [24] a design pattern for representing them is proposed. However, as it is defined as an extension of the *Collection* concept, the proposal introduces many concepts and relations that are not strictly necessary, lowering the performance of reasoners. Concepts such as *item* or *Bag* and relations such as *'has member'* are not necessary, for example. It also defines the concept *Collection* to be disjoint with the concept *item*, so it is not possible to represent lists having other lists as items. This makes it impossible to define lists of activities.

Two general design patterns for representing lists are also proposed in [23]. Our proposal is similar to the pattern that models lists directly as chains of individuals but, for the sake of clarity and standardization, we opted for using the properties names in the pattern that models lists as data structures. We have also introduced some differences in our proposal for representing activities as lists of events, which will be explained in detail bellow.

Let $\mathscr{L}$ be an activity comprising a set of events $\{e_1, e_2, \ldots, e_n\}$ and $<$ a strict partial order, i.e. a binary relation that is irreflexive, transitive and asymmetric, defined for each pair $\langle e_i, e_j \rangle \in \mathscr{L} \times \mathscr{L}$, where $\times$ is the cartesian product. Based on the previous definitions, the concepts $Activity \sqsubseteq \top$ and $Event \sqsubseteq \top$ are defined, as well as the following two properties:

$$hasNext \sqsubseteq isFollowedBy$$

hasNext is defined as a functional, asymmetric and irreflexive property, establishing the order of events in the activity $\mathscr{L}$ according to the $<$ order. Due to the fact that it has been defined as a functional property just one event could follow another event. The inverse property is also defined as functional, forcing an event to be directly preceded by a unique event. The full definition of the $<$ order is achieved by introducing the transitive property *isFollowedBy* as a superproperty of *hasNext*. Since this means that *hasNext* implies *isFollowedBy*, any sequence of entities linked by *hasNext* will be inferred to be a chain linked by *isFollowedBy*. *hasNext* is used to express that an event *B* immediately follows another event *A*. There is no other event between them. So event *A* has *B* as the next event in the list (*A hasNext B*) or, in other words, event *A* is followed by event *B* (*A isFollowedBy B*). If another event *C* appears after event *B*, event *A* is also followed by event *C* (*A isFollowedBy C*), but event *A* has not *C* as the next event on the list (*not A hasNext C*).

The property *hasItem* establishes the membership of an event to the list. The class description *hasItem some (Frontdoor_set and* *isFollowedBy some Dishwasher_set)* is a way of describing the activity *#Activity24* of the example in Fig. 2.

The properties *startsWith* and *endsWith* are used to identify the first and last events in the activities. Due to open world assumption in OWL, reasoners cannot automatically infer the individuals that belong to these concepts. Therefore, it is necessary to annotate these individuals when the activities are converted to the model proposed in this paper. The class description *startsWith some (Frondoor_set and hasNext some (Fridge_set and hasNext some Dishwasher_set))*, for example, represents the activities that begin with the activation of the sensor of the front door, which is immediately followed by the activation of the fridge sensor and then by the dishwasher sensor, immediately after. The activity *#Activity24* in Fig. 2 is an example of activity described by the above class description.

In OWL the same individual could be referred to in many different ways (i.e. with different URI references). Due to this, it is necessary to state that all the elements in the datasets are different individuals. For practical reasons, a functional property *hasID* is used to identify all of the individuals in the model with a unique code. In this way, the addition of new entities to the ontology is easier, without the need for asserting that all of them are different from the existing individuals.

$$\top \sqsubseteq \forall\ hasID\ Datatype\#long$$

Due to the high formalization of ontologies, it is not necessary to make all relations in the datasets explicit. Many of them may be inferred by the reasoner. Knowing that *#Event15 hasNext #Event16*, the reasoner may infer that *#Event15 isFollowedBy #Event16*, since *hasNext*$\sqsubseteq$*isFollowedBy*. In addition, if *#Event16 hasNext #Event17*, the reasoner may easily infer that *#Event15 isFollowedBy #Event17*, since the property *isFollowedBy* has been declared as transitive.

When we want to refer to events that occur before another one we can make use of inverse properties, which have not been explicitly defined for efficiency reasons. *#Activity24* may also be described by the class description *endsWith hasNext*$^{-}$ *Fridgeset*, which describes activities ending with an event preceded by an activation of the fridge sensor.

3.2. Functional architecture

All the components of the methodology proposed in this paper and how the information flows among them are explained in this section and depicted in Fig. 3.

The system starts from a dataset with a set of labeled activities. First of all it is necessary to convert the dataset information into an ontology. For this, we have developed an application that first translates the information in the dataset to a common data model in XML format. The XML file contains all the events produced by sensors for each activity, as well as other information such as the sensors that were active during the activity or the type of the sensors. The application is currently capable of loading datasets with the formats proposed in [14,15].

A second application is responsible for the transformation of the activities expressed in the XML format to an ontology that uses the primitives described in Section 2. In this step two text files are also generated that contain: a) the list of individuals in the ontology of the kind of activity to be recognized by the classifier (positives), and b) the rest of individuals (negatives). These lists of individuals will be used to generate the input data for the classifier in a later step.

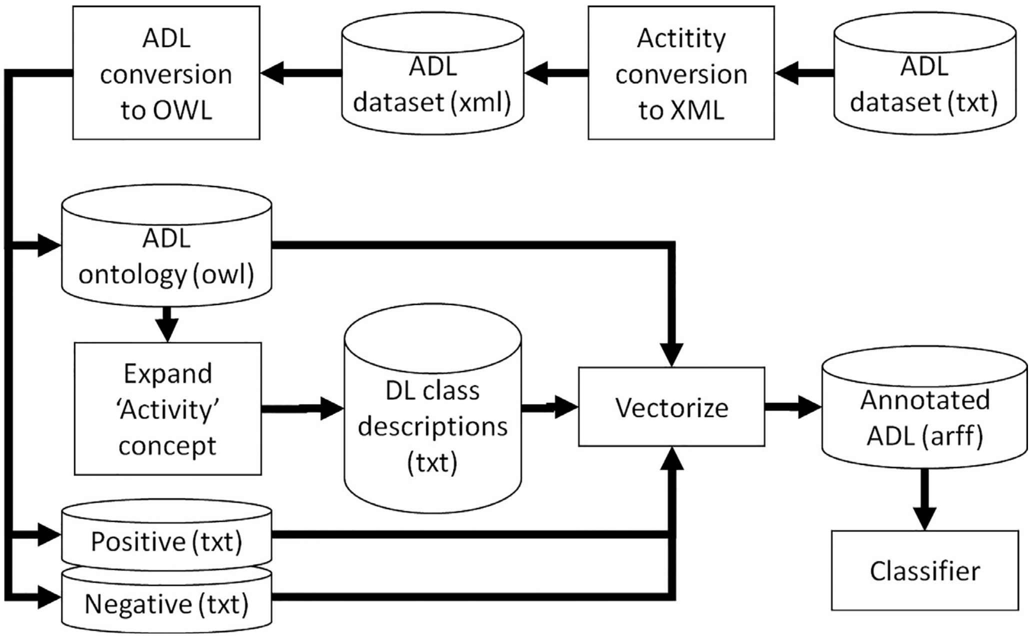


Fig. 3. Functional architecture.

Next, it is necessary to expand the definition of the *Activity* concept in the ontology. The expansion process consists of generating new class descriptions that represent different patterns of activities, without taking into account the specific type of activity that is going to be recognized by the classifier. More specifically, another application takes the concept to be expanded (*Activity*) as an argument and uses a given set of classes, properties and operators to construct new class descriptions in DL. All the concepts in the ontology that at least describe some individual in the ontology have been taken as the set of classes L. All properties defined in the ontology have been taken as the set of properties P. The set of operators O is specified by the user and consists of a subset of all operators that can be used to combine class descriptions (see Section 2.2).

The expansion process begins by combining all the concepts in L by means of O operators. The complement operator ($\mathscr{C}$) results in class expressions of the form $not\, c_i$, where $c_i \in L$. Class expressions such as *not Cupboard_set* or *not Activity* are produced using the complement operator, for example.

All class descriptions in L are combined with themselves in the case of operators that require two class descriptions, resulting in expressions of the form $c_i\, o_k\, c_j$, where $c_i,\ c_j \in L,\ i \neq j$ and $o_k \in \{and,\ or\}$. In this process, expressions such as *Event and Cupboard_set* or *Cupboard_set or HallBedroom_Door_set* are generated, for example.

There are operators that require a property to form valid class descriptions. They are the existential quantifiers and the universal quantifier. In this case, the expansion process combines all class descriptions in L with all properties in the ontology, producing expressions of the form $p_i\, o_k\, c_j$, where $p_i \in P,\ o_k \in \{some,\ all\}$ and $c_j \in L$. *startsWith some HallBedroom_Door_set* or *isFollowedBy all Cupboard_set* are examples of expressions generated by existential and universal quantifiers. These class expressions represent all those individuals that begin with the firing of the *HallBedroom_Door* sensor and all those individuals that are only followed by *Cupboard* sensor activations, respectively.

The last type of operators that implements the application are the cardinality constraints. These operators limit the number of individuals to which an individual may be related among a given property. The class descriptions generated with these operators have the form $p_i\, or_k\, n\, c_j$, where $p_i \in P,\ o_k \in \{min,\ max,\ exact\},\ c_j \in L$ and $n \in N$. Expressions such as *isFollowedBy min 4 Event* or *isFollowedBy exact 2 Cupboard_set* are generated, for example, representing the set of individuals followed by at least four sensor activations and the set of individuals followed by exactly two activations of the *Cupboard* sensor, respectively. The number of constraints to be generated is virtually infinite since $n \in N$. It is the user who must specify the possible values for n. For example, $n \in \{2, 3\}$ in the experiment of Section 4.

All the expressions generated are added to L and the process is repeated again. However, not all of the expressions generated are relevant. Some of them are simply unsatisfiable. A class expression such as *hasItem some Activity*, for example, is unsatisfiable since the range of the property *hasItem* is the *Event* concept, which is defined to be disjointed with the concept *Activity*. There cannot be an individual in the ontology which meets such restriction. For the same reason, expressions like *hasItem some startsWith some HallBedroom_Door_set* are also unsatisfiable, since the domain of the *startsWith* property is the concept *Activity*. Only satisfiable class expressions are added to L.

On the other hand, not all class expressions in L describe activities. With the help of the reasoner, a new set $V \subseteq L$ is created, which contains all the class expressions in L that describe activities. These are the expressions that the application produces as result, in a text file.

The last application takes the class expressions generated in the previous step as input and produces a table with k rows and n binary columns, where k is the number of labeled activities in the dataset and n is the number of class descriptions generated in the expansion process. Each of the rows is therefore a vector $F^k = \left\{f_1^k, \ldots, f_j^k, \ldots, f_n^k, f_{n+1}^k\right\}$. Each of the n generated class expressions corresponds to a feature $f_j^k \in F^k$. $F_j^k = 1$ if the activity k is an instance of the class description j. $F_j^k = 0$ otherwise. $F_{n+1}^k = 1$ if the activity k is an instance of the kind of activity to be recognized by the classifier (positive). $F_{n+1}^k = 0$ otherwise. The list of labeled individuals generated at the beginning of the process is used for this purpose. This process is called "vectorization". An example of the results obtained by this application is shown in Table 2.

4. Experiments

In order to evaluate the quality of the methodology proposed in this work an experiment has been carried out, in which the datasets proposed in [14–16] have been used. In the case of Ordónez et al. [16], two sets of activities and sensors are presented, so they are considered as two different datasets. The objective of the experiment is to determine whether or not a particular ADL has been performed based on the sensors that have been fired during a specific period of time. To simplify the experiment, the time

Table 2
Example of data produced by the 'vectorization' process.

Activity	*startsWith some HallBedroom_Door_set*	*hasItem min 2 Hall-Bedroom_door_set*	Positive
1	1	0	1
2	0	1	0
3	1	1	1
4	0	0	1

Table 3
Activity recognition accuracy for classic approach.

Dataset	Activity	C4.5	SMO	VP	DT
Singla	Answer the phone	99.40	99.40	89.39	99.40
Singla	Choose outfit	100.00	100.00	96.64	100.00
Singla	Clean	96.03	**98.41**	96.81	96.83
Singla	Fill medication	100.00	100.00	96.63	100.00
Singla	Prepare birthday card	98.80	99.40	97.43	98.80
Singla	Prepare soup	100.00	100.00	99.00	100.00
Singla	Watch DVD	99.01	100.00	96.58	97.61
Singla	Water plants	97.59	99.41	96.21	97.59
Kasteren	Get drink	97.70	98.91	94.97	96.46
Kasteren	Go to bed	94.16	94.67	91.32	94.69
Kasteren	Leave house	100.00	100.00	98.78	100.00
Kasteren	Prepare breakfast	97.00	96.18	94.83	96.32
Kasteren	Prepare dinner	96.59	97.68	96.05	96.06
Kasteren	Take shower	97.56	97.96	93.62	90.63
Kasteren	Use toilet	**91.97**	91.16	89.39	90.09
Ordoñez (a)	Breakfast	99.59	99.59	98.91	99.59
Ordoñez (a)	Dinner	100.00	100.00	100.00	100.00
Ordoñez (a)	Grooming	95.72	95.72	95.31	95.98
Ordoñez (a)	Leaving	99.59	99.59	99.06	99.59
Ordoñez (a)	Lunch	100.00	100.00	97.04	100.00
Ordoñez (a)	Showering	100.00	100.00	99.46	100.00
Ordoñez (a)	Sleeping	100.00	100.00	99.73	100.00
Ordoñez (a)	Snack	98.66	100.00	95.70	98.26
Ordoñez (a)	Spare time TV	96.49	97.16	97.58	97.16
Ordoñez (a)	Toileting	**94.33**	93.93	86.95	89.49
Ordoñez (b)	Breakfast	94.87	95.00	95.34	94.73
Ordoñez (b)	Dinner	97.77	97.77	97.77	97.77
Ordoñez (b)	Grooming	95.05	95.39	94.11	95.12
Ordoñez (b)	Leaving	99.19	99.53	99.12	98.92
Ordoñez (b)	Lunch	97.37	97.37	97.30	97.17
Ordoñez (b)	Showering	100.00	100.00	99.53	100.00
Ordoñez (b)	Sleeping	99.39	99.39	98.45	99.39
Ordoñez (b)	Snack	**91.96**	90.13	90.20	90.61
Ordoñez (b)	Spare time TV	95.74	95.74	95.53	95.74
Ordoñez (b)	Toileting	98.39	98.39	98.39	98.39

intervals always correspond to the labeled activities in the dataset.

The results obtained by four classifiers that use a classic DDA to solve this problem have been taken as reference to measure the efficiency of our proposal. For this purpose an application that identifies the sensors that have been fired during each of the activities has been built. The application generates a file in Weka format, following the structure presented in Section 2.1. This file contains an instance for each activity and as many features as sensors in the dataset. All the features are binary and specify if the sensor has been fired during the activity or not. Finally, it includes a class attribute, also binary, that indicates if it is the activity that the classifier is learning to identify (positive) or not (negative). Each experiment consists, therefore, of determining which combination of sensors are fired for a particular activity, such as “take shower”, for example.

By using the Weka data mining software [25], we have generated C4.5,[1] Sequential Minimal Optimization (SMO), Voted perceptron (VP) and Decision Table (DT) classifiers for all the activities in the datasets. A summary of the results obtained for all the activities of the different datasets is shown in Table 3. The most difficult activities to be recognized using these algorithms have been taken as reference. Specifically, the activity chosen for the experiment from the dataset in [15] has been the activity “Clean”, with an accuracy of 98.41% at best. The activity chosen from Van Kasteren et al. [14] has been “Use toilet”, with 92.40% accuracy. The activity chosen from the first dataset in [16] has been “Toileting”, while we have selected the activity “Snack” from the second dataset.

For each of the datasets a new ontology has been automatically generated using the primitives in Section 3.1. Another application is then used to generate a specific number of new class descriptions for each of them. All new class descriptions describe the *Activity* concept. Three subsets of operators have been used to generate three different sets of new class descriptions. For the first one, all available operators ($\mathscr{ACIXMSU}$) have been used. The complement, minimum cardinality and the existential quantifier operators ($\mathscr{CMS}$) have been used for the second one and only the existential quantifier ($\mathscr{S}$) has been used for the latter. Versions with 50, 100, 150 and 200 class expressions have been generated for each of these sets. Versions with up to 500 class expressions have also been generated for the cases where only the existential quantifier operator has been used.

All files with new class descriptions are evaluated by another application and a new file in Weka format is generated for each of

[1] The Weka implementation of the C4.5 classifier is called J48.

Table 4
"Use toilet" classification performance.

Dataset	$\|\mathcal{F}\|$	$\|\mathcal{F}^*\|$	%	Expand	Vectorize	Total	C4.5	SMO	VP	DT	Best
Classic	14	14	100.00				**91.97**	91.16	89.39	90.09	91.97
$\mathcal{ACIXMSU}$	50	46	92.00	1.38	73.05	74.43	86.53	87.33	84.88	85.03	87.33
$\mathcal{ACIXMSU}$	100	94	94.00	9.53	69.04	78.57	95.77	**96.71**	91.78	94.80	96.71
$\mathcal{ACIXMSU}$	150	143	95.33	33.01	94.50	127.51	95.77	96.57	91.69	94.66	96.57
$\mathcal{ACIXMSU}$	200	194	97.00	78.79	682.29	761.08	95.77	96.03	92.76	94.79	96.03
$\mathcal{CMS}$	50	46	92.00	0.75	79.81	80.56	86.53	87.33	84.88	85.03	87.33
$\mathcal{CMS}$	100	91	91.00	0.94	240.05	240.99	95.77	95.76	90.33	94.53	95.77
$\mathcal{CMS}$	150	134	89.33	1.12	1159.07	1160.19	95.22	96.30	90.44	93.71	96.30
$\mathcal{CMS}$	200	178	89.00	1.31	1513.29	1514.60	95.22	**96.98**	90.62	93.71	96.98
$\mathcal{S}$	50	44	88.00	0.70	1.27	1.97	96.32	96.32	91.28	95.76	96.32
$\mathcal{S}$	100	77	77.00	0.81	1.95	2.76	95.62	**97.67**	90.87	94.81	97.67
$\mathcal{S}$	150	109	72.67	0.93	2.40	3.33	97.26	97.53	92.76	94.96	97.53
$\mathcal{S}$	200	142	71.00	1.00	3.10	4.10	97.26	96.99	91.02	94.96	97.26
$\mathcal{S}$	250	174	69.60	1.16	3.48	4.64	97.26	97.39	90.46	94.96	97.39
$\mathcal{S}$	300	207	69.00	1.28	3.93	5.21	97.26	97.53	89.26	94.96	97.53
$\mathcal{S}$	350	240	68.57	1.53	4.61	6.14	97.26	97.26	89.77	94.96	97.26
$\mathcal{S}$	400	273	68.25	1.54	4.65	6.19	97.26	97.39	87.98	94.96	97.39
$\mathcal{S}$	450	307	68.22	1.70	5.42	7.12	97.26	96.86	88.96	94.96	97.26
$\mathcal{S}$	500	340	68.00	1.77	5.51	7.28	97.26	96.44	90.07	94.96	97.26

them. The same four types of classifiers that were employed to evaluate the performance of the classifiers using the classical approach have been used to evaluate the accuracy of the classifiers based on the new approach. Results are discussed in the next section.

5. Results

The accuracy obtained by all the different DDA for the selected activities are analyzed in this section. First, the results obtained for the four selected activities of the different datasets are analyzed in detail. At the end of the section, there is a general analysis of the results obtained for all the activities available in the three datasets of the experiment.

Table 4 shows the results obtained by the classifiers for the activity "Clean" in [14]. The first column indicates the set of operators used to generate the features for the classifiers. The second column shows the number of features ($|\mathcal{F}|$) that have been generated in the expansion process. Some of them have been removed from the final set of features because they are not relevant. A feature is not considered relevant if all the instances have the same value for it. The third column contains the final number of features available for the classifiers ($|\mathcal{F}^*|$). The total number of sensors in the dataset have been used for the classic approach in both columns. The percentage of relevant features considered by the classifier is indicated in the fourth column (%). The next two columns show the time spent in the processes of expansion and vectorization, in seconds. The sum of both values is shown in the column 'Total'. The next four columns indicate the accuracy obtained for the C4.5, SMO, VP and DT classifiers, in percentage values. The best results for each approach are in bold. The accuracy of the best classifier is finally shown in the last column of the table.

The first row contains the data related to the classifiers constructed using the classical approach. This approach yields a precision of 91.97% for the best case of the activity 'Use toilet'. In spite of being a high value, most of the classifiers created with the approach proposed in this work widely surpassed that value. In fact, the best accuracy (97.67%) is obtained for the SMO classifier that only use the existential quantifier for generating the sets of DL class expressions. Only one hundred of them are needed to obtain such precision value. It is a statistically significant improvement against the results obtained by the classic approach for a confidence of $p = 0.05$ (two tailed). To check this, we have employed the *Paired T-Tester* test of Weka. The *corrected* version of the tester has been used because we are using cross validations in the experiments. The classifiers that use sets of expressions formed by more operators also achieve very good results, but somewhat lower. In addition, for classifiers using a datasets of type $\mathcal{CMS}$, two hundred class expressions are required to obtain the best results.

It is worth noting the significant difference with respect to the time used by the classifiers depending on the set of operators used to generate the class expressions. Classifiers that use datasets of type $\mathcal{S}$ and $\mathcal{CMS}$ require just one second to generate two hundred class expressions, whereas in the case of the classifier based on the datasets of type $\mathcal{ACIXMSU}$ it takes 78.79 s to generate the same number of class descriptions. The difference becomes much more noticeable when evaluating the time spent in the process of vectorization. The classifier that only uses the existential quantifier to generate class expressions only takes 3.10 s to evaluate the two hundred class expressions generated, whereas more than twenty-five minutes are needed to evaluate the same number of expressions in the case of the dataset of type $\mathcal{CMS}$. Although it contains more complex expressions, the time required to evaluate the class expressions contained in the dataset of type $\mathcal{ACIXMSU}$ is just eleven minutes. This is because in the dataset of type $\mathcal{CMS}$ there are many more class expressions that contain cardinality restrictions, which require much more time to be computed by the reasoner.

The behavior of the classifiers developed for the activity "Snack (b)", from the dataset in [16], is very similar to the activity "Use toilet" discussed above. In this case, the improvement over the classifiers based on a classical approach is 2.7 points, while in the case of the activity "Use toilet" the improvement was 5.7 points. The best value (94.38%) achieved by the SMO algorithm is statistically significant with respect to classic approach (90.13%) for a confidence of $p = 0.05$ while the best value (94.66) achieved for the C4.5

Table 5
"Snack (b)" classification performance.

Dataset	$\|\mathcal{F}\|$	$\|\mathcal{F}^*\|$	%	Expand	Vectorize	Total	C4.5	SMO	VP	DT	Best
Classic	10	10	100.00				**91.96**	90.13	90.20	90.61	91.96
$\mathcal{ACIXMSU}$	50	41	82.00	1.21	45.64	46.85	94.25	94.38	90.47	90.47	94.38
$\mathcal{ACIXMSU}$	100	83	83.00	10.16	125.86	136.02	94.25	93.98	90.67	90.47	94.25
$\mathcal{ACIXMSU}$	150	129	86.00	27.99	171.33	199.32	**94.66**	93.84	90.40	90.47	94.66
$\mathcal{ACIXMSU}$	200	173	86.50	48.59	206.37	254.96	94.66	93.64	90.47	90.47	94.66
$\mathcal{CMS}$	50	41	82.00	0.81	49.73	50.54	94.25	93.91	90.47	90.53	94.25
$\mathcal{CMS}$	100	82	82.00	0.97	138.28	139.25	93.71	93.64	90.40	91.88	93.71
$\mathcal{CMS}$	150	115	76.67	1.13	358.95	360.08	93.71	93.64	90.74	92.15	93.71
$\mathcal{CMS}$	200	154	77.00	1.27	689.71	690.98	94.25	**94.38**	90.47	90.47	94.38
$\mathcal{S}$	50	42	84.00	0.77	2.00	2.77	91.21	90.39	90.27	90.20	91.21
$\mathcal{S}$	100	70	70.00	0.86	2.38	3.24	90.67	89.79	90.27	89.45	90.67
$\mathcal{S}$	150	100	66.67	0.98	3.69	4.67	89.93	89.92	90.27	89.92	90.27
$\mathcal{S}$	200	131	65.50	1.02	4.21	5.23	89.79	91.41	89.93	89.52	91.41
$\mathcal{S}$	250	162	64.80	1.25	4.73	5.98	89.59	91.28	90.40	89.39	91.28
$\mathcal{S}$	300	188	62.67	1.32	5.06	6.38	91.95	91.95	90.27	91.27	91.95
$\mathcal{S}$	350	218	62.29	1.34	5.79	7.13	91.95	91.81	90.47	91.07	91.95
$\mathcal{S}$	400	247	61.75	1.46	6.79	8.25	91.88	91.88	90.40	91.20	91.88
$\mathcal{S}$	450	278	61.78	1.50	7.05	8.55	91.88	91.95	90.47	91.47	91.95
$\mathcal{S}$	500	312	62.40	1.80	7.32	9.12	**93.84**	92.76	90.40	92.96	93.84

algorithm is significant for a confidence of $p = 0.1$ (Table 5).

There is also a significant decrease in the time required to vectorize the datasets of type $\mathcal{CMS}$ and $\mathcal{ACIXMSU}$. It also happens that in this case the results are improved as the number of operators used to generate the sets of class expressions increases. In fact, classifiers that use datasets of type $\mathcal{S}$ do not achieve the same accuracy as classifiers using the datasets of type $\mathcal{ACIXMSU}$ even after generating five hundred class expressions from the dataset. In contrast, it should also be noted that only 9.12 s are required to process those five hundred class expressions, whereas for the dataset of type $\mathcal{CMS}$, it takes more than three minutes to process two hundred of them.

Table 6 shows the results obtained by the classifiers for the activity "Clean" in [15]. The classifiers constructed using the classical approach yields a precision of 98.41% for the best case. In spite of being a very high value, some of the classifiers created with the approach proposed in this work slightly surpassed that value. In fact, the same accuracy (98.81%) is obtained for the three different types of datasets generated. Two hundred of them are needed at least to obtain such precision value for the datasets that contain more complex expressions. Only one hundred and fifty class expressions are needed for the dataset that only contains expression generated using the existential quantifier. No statistical significance has been found in this case, mainly due to the high precision value that is taken as reference.

On the other hand, it is worth noting the significant difference with respect to the time used by the classifiers depending on the set of operators used to generate the datasets. The classifier that uses datasets of type $\mathcal{CMS}$ requires 2.27 s to generate two hundred class expressions, while the classifier that uses datasets of type $\mathcal{S}$ requires the same time to generate double the amount of class

Table 6
"Clean" classification performance.

Dataset	$\|\mathcal{F}\|$	$\|\mathcal{F}^*\|$	%	Expand	Vectorize	Total	C4.5	SMO	VP	DT	Best
Classic	45	45	100.00				96.03	**98.41**	96.81	96.83	98.41
$\mathcal{ACIXMSU}$	50	37	74.00	5.97	42.30	48.27	94.79	97.62	90.82	94.79	97.62
$\mathcal{ACIXMSU}$	100	72	72.00	9.01	100.50	109.51	97.01	96.80	92.98	97.22	97.22
$\mathcal{ACIXMSU}$	150	110	73.33	11.85	201.75	213.60	97.22	97.19	91.84	96.42	97.22
$\mathcal{ACIXMSU}$	200	141	70.50	14.29	986.49	1000.78	97.22	**98.81**	91.64	95.22	98.81
$\mathcal{CMS}$	50	37	74.00	1.52	42.94	44.46	94.79	97.62	90.82	94.79	97.62
$\mathcal{CMS}$	100	72	72.00	1.84	98.18	100.02	97.01	96.80	92.98	97.22	97.22
$\mathcal{CMS}$	150	110	73.33	1.97	195.32	197.29	97.22	97.19	91.84	96.42	97.22
$\mathcal{CMS}$	200	141	70.50	2.27	990.35	992.62	97.22	**98.81**	91.64	95.22	98.81
$\mathcal{S}$	50	33	66.00	1.32	3.43	4.75	94.39	97.18	95.43	93.42	97.18
$\mathcal{S}$	100	69	69.00	1.43	5.77	7.20	96.00	96.61	95.22	94.00	96.61
$\mathcal{S}$	150	93	62.00	1.67	7.17	8.84	95.82	**98.81**	96.81	96.20	98.81
$\mathcal{S}$	200	121	60.50	1.76	8.94	10.70	95.82	97.99	97.40	96.20	97.99
$\mathcal{S}$	250	151	60.40	2.02	10.56	12.58	95.82	98.20	97.21	96.20	98.20
$\mathcal{S}$	300	183	61.00	2.12	12.04	14.16	95.82	97.78	96.97	96.20	97.78
$\mathcal{S}$	350	212	60.57	2.20	13.14	15.34	95.82	97.59	96.79	96.20	97.59
$\mathcal{S}$	400	240	60.00	2.31	15.31	17.62	95.82	98.20	96.21	96.20	98.20
$\mathcal{S}$	450	274	60.89	2.61	16.66	19.27	95.82	98.20	97.22	96.20	98.20
$\mathcal{S}$	500	305	61.00	2.75	17.87	20.62	95.82	97.99	96.81	96.20	97.99

Table 7
"Toileting (a)" classification performance.

Dataset	$\|\mathcal{F}\|$	$\|\mathcal{F}^*\|$	%	Expand	Vectorize	Total	C4.5	SMO	VP	DT	Best
Classic	12	12	100.00				**94.33**	93.93	86.95	89.49	94.33
$\mathcal{ACIXMSU}$	50	37	74.00	1.37	1.53	2.90	86.56	90.99	84.56	86.70	90.99
$\mathcal{ACIXMSU}$	100	77	77.00	10.53	2.87	13.40	**94.76**	94.36	87.08	94.49	94.76
$\mathcal{ACIXMSU}$	150	117	78.00	35.02	4.51	39.53	94.76	94.36	88.44	94.76	94.76
$\mathcal{ACIXMSU}$	200	159	79.50	72.06	5.64	77.70	94.76	94.36	88.97	94.76	94.76
$\mathcal{CMS}$	50	37	74.00	0.61	1.46	2.07	86.56	90.99	84.56	86.70	90.99
$\mathcal{CMS}$	100	71	71.00	0.82	4.98	5.80	**94.76**	94.22	87.24	94.76	94.76
$\mathcal{CMS}$	150	107	71.33	0.99	10.59	11.58	94.76	94.22	87.09	94.76	94.76
$\mathcal{CMS}$	200	143	71.50	1.21	22.20	23.41	94.76	94.22	86.83	94.76	94.76
$\mathcal{S}$	50	45	90.00	0.58	0.83	1.41	**94.76**	94.76	89.01	94.62	94.76
$\mathcal{S}$	100	70	70.00	0.69	1.06	1.75	94.76	94.36	85.35	94.62	94.76
$\mathcal{S}$	150	99	66.00	0.80	1.40	2.20	94.76	94.36	84.13	94.76	94.76
$\mathcal{S}$	200	127	63.50	0.90	1.48	2.38	94.76	94.36	83.19	94.76	94.76
$\mathcal{S}$	250	156	62.40	0.98	1.67	2.65	94.76	94.36	81.99	94.76	94.76
$\mathcal{S}$	300	166	55.33	1.19	1.96	3.15	94.76	94.36	82.27	94.76	94.76
$\mathcal{S}$	350	185	52.86	1.28	2.10	3.38	94.76	94.36	82.13	94.76	94.76
$\mathcal{S}$	400	196	49.00	1.42	2.34	3.76	94.76	94.36	82.13	94.76	94.76
$\mathcal{S}$	450	212	47.11	1.45	2.38	3.83	94.76	94.36	82.13	94.76	94.76
$\mathcal{S}$	500	234	46.80	1.58	2.63	4.21	94.76	94.36	82.13	94.76	94.76

descriptions. In order to generate the same number of class expressions, the classifier that uses datasets of type $\mathcal{ACXMSU}$ requires 14.29 s, a value six times higher. The difference becomes much more noticeable when evaluating the time spent in the process of vectorization. The classifier that only uses the existential quantifier to generate class expressions only takes 17.87 s to evaluate five hundred class expressions, whereas the other kind of classifiers take more than fifteen minutes to evaluate two hundred class expressions.

The same behavior is observed for the activity "Toileting (a)" (Table 7) from the dataset in [16]. The only notable difference is the significant decrease in the time needed to process the datasets, which is mainly due to the difference in size of the datasets. The ontology produced by the transformation of the dataset in [15] contains 17,427 logical axioms and 5676 individuals, whereas the ontology produced by transformation of the first dataset in [16] contains only 1602 logical axioms and 613 individuals. Only in the case of datasets of type $\mathcal{ACIXMSU}$ more than a minute is required to process them. In the worst case, the optimum value is reached at 13.40 s. In the best case, only 1.41 s and fifty class expressions are required to reach the optimal value ($\mathcal{S}$).

Regarding the accuracy of the classifiers with respect to the set of operators used in this experiment, it is noteworthy that there is no significant difference. The best values are sometimes obtained for the classifiers using datasets of type $\mathcal{ACIXMSU}$, sometimes using the datasets of type $\mathcal{CMS}$ and sometimes using the dataset of type $\mathcal{S}$. However, there is a significant difference in the time required to generate and evaluate the class descriptions in the different datasets. In the activities selected for the analysis in detail these times vary between the 1514.60 s needed to generate two hundred class expressions for the dataset of type $\mathcal{CMS}$ and the 10.70 s required for the type $\mathcal{S}$. In the case of the datasets of types $\mathcal{ACXMSU}$ and $\mathcal{CMS}$ class expressions such as *"hasItem min 2 Hall-Bedroom_door_set"* are generated, for example, representing those activities in which the *Hall-Bedroom_door* sensor is activated at least twice during the activity. In spite of the apparent simplicity of this class expression, the time required by the reasoner to determine the activities represented by that class expression is very high. All activities that include cardinality constraints require a long period of time to be evaluated. The classifiers that use datasets of type $\mathcal{CMS}$ produces more class expressions with cardinality constraints, so they take more time than the classifiers that use other types of dataset.

The class expressions generated by the classifiers using datasets of type $\mathcal{S}$ include, for instance, expressions such as *"hasItem some (isFollowedBy some Hall-Toilet_door_set)"* or *"startsWith some (hasNext some Hall-Bathroom_door_set)"*, which represent activities in which a sensor is fired before the *Hall-Toilet_door* sensor and activities in which the sensor *Hall-Bathroom_door* is the second sensor to be fired, respectively. It is worth mentioning that the class expression *"hasItem min 2 Hall-Bedroom_door_set"* is equivalent to the expression *"hasItem some (Hall-Bedroom_door_set and (isFollowedBy some Hall-Bedroom_door_set))"* but the latter requires much less time to be computed than the former.

Regarding the type of classifiers used, after this first analysis it can be verified that in general the best results are obtained for classifiers of type C4.5 and SMO. Classifiers of type VP and DT only obtain the best results in very few cases. Anyway, one of the advantages offered by the system proposed in this work is the possibility of identifying the relevant features for the classifiers. However, the VP classifier is the only classifier of the four employed in the test that does not allow the identification of these features, because it is a 'black box' classifier.

Table 8 summarizes the best accuracy obtained for the classifiers generated for all the activities in the datasets. The third column shows the accuracy of the best classifier that uses a classic approach. The fourth column shows the accuracy of the best classifier that uses the proposed approach. The difference between those values is shown in the sixth column and, finally, the last column indicates the percentage of gain achieved by the classifier with respect to the maximum possible gain. As can be seen, the classifiers based on the proposal presented in this paper always improve or, in the worst case, match the accuracy obtained by the classifiers using the classic approach. The average classification accuracy for the classifiers using the classic approach is 98.08. The average classification

Table 8
Global classification accuracy.

Dataset	Activity	Classic	Proposal	Gain	% Gain
Singla	Answer the phone	99.40	99.41	0.01	1.67
Singla	Choose outfit	100.00	100.00	0.00	
Singla	Clean	98.41	98.81	0.40	25.16
Singla	Fill medication	100.00	100.00	0.00	
Singla	Prepare birthday card	99.40	100.00	0.60	100.00
Singla	Prepare soup	100.00	100.00	0.00	
Singla	Watch DVD	100.00	100.00	0.00	
Singla	Water plants	99.41	99.39	−0.02	−3.39
Kasteren	Get drink	98.91	99.59	0.68	62.39
Kasteren	Go to bed	94.67	99.06	4.39	82.36
Kasteren	Leave house	100.00	100.00	0.00	
Kasteren	Prepare breakfast	97.00	98.50	1.50	50.00
Kasteren	Prepare dinner	97.68	99.60	1.92	82.76
Kasteren	Take shower	97.96	99.59	1.63	79.90
Kasteren	Use toilet	91.97	97.67	5.70	70.98
Ordoñez (a)	Breakfast	99.59	99.59	0.00	0.00
Ordoñez (a)	Dinner	100.00	100.00	0.00	
Ordoñez (a)	Grooming	95.98	96.64	0.66	16.42
Ordoñez (a)	Leaving	99.59	100.00	0.41	100.00
Ordoñez (a)	Lunch	100.00	100.00	0.00	
Ordoñez (a)	Showering	100.00	100.00	0.00	
Ordoñez (a)	Sleeping	100.00	100.00	0.00	
Ordoñez (a)	Snack	100.00	100.00	0.00	
Ordoñez (a)	Spare time TV	97.58	98.98	1.40	57.85
Ordoñez (a)	Toileting	94.33	94.76	0.43	7.58
Ordoñez (b)	Breakfast	95.34	97.50	2.16	46.35
Ordoñez (b)	Dinner	97.77	97.97	0.20	8.97
Ordoñez (b)	Grooming	95.39	97.30	1.91	41.43
Ordoñez (b)	Leaving	99.53	99.80	0.27	57.45
Ordoñez (b)	Lunch	97.37	98.11	0.74	28.14
Ordoñez (b)	Showering	100.00	100.00	0.00	
Ordoñez (b)	Sleeping	99.39	99.39	0.00	0.00
Ordoñez (b)	Snack	91.96	94.66	2.70	33.58
Ordoñez (b)	Spare time TV	95.74	95.75	0.01	0.23
Ordoñez (b)	Toileting	98.39	98.38	−0.01	−0.62
Average		98.08	98.87	0.79	41.18

accuracy for the classifiers using the proposed approach is 98.87. This mean that the classifiers using the proposed approach have trimmed down the difference with respect to the perfect classifiers a 41.18%.

As pointed out in Section 2.3, CEL is the most similar technique to our proposal. We have conducted a small experiment to test it with three activities of the Ordoñez (b) dataset. The prediction accuracies reported by the DL-Learner application,[2] have been 95.74%, 97.57% and 95.54% for the “Snack”, “Breakfast” and “Grooming” activities, respectively. Apparently, the CEL approach performs slightly better than our proposal for two of the three activities. However, it should be noted that DL-Learner makes use of its own approximate incomplete reasoning procedure for Fast Instance Checks which partially follows a closed world assumption. This means that the results produced by the DL-Learner application are not the same as those provided by reasoners that complies with the OWL standard, so the produced class expressions only make sense in the context of the DL-Learner application. As an example, the following class expression is the one that DL-Learner found to be the best description for the “Breakfast” activity.

$$\begin{aligned}&\mathit{hasItem}\ \mathit{min}\ 4\ (\mathit{Fridgeset}\ \mathit{or}\ \mathit{Microwaveset}\ \mathit{or}\ (\mathit{DoorKitchenset}\ \mathit{and}\\&\qquad(\mathit{isFollowedBy}\ \mathit{only}\ (\mathit{not}\ (\mathit{DoorLivingset})))))\end{aligned}$$

However, when the expression is evaluated by the HermiT reasoner,[3] only four instances are found for the activity, two of them being incorrectly classified. No instances were found for the “Snack” activity.

6. Conclusion and future works

This paper has been focused on a new methodology that uses ontology for the purpose of sensor-based activity recognition with DDA in order to increase the accuracy in the classification process. To do so, the set of feature vectors computed by the dataset are extended with the asserted and the inferred knowledge from the ontology that describe the dataset itself. An evaluation has been carried out with the following four popular classifiers: C4.5, Sequential Minimal Optimization, Voted perceptron and Decision Table.

[2] http://dl-learner.org.
[3] http://www.hermit-reasoner.com.

Results from the evaluation demonstrated the ability of the ontology to extend the vector features to provide an increase of the performance in all evaluated classifiers.

However, there is still room for the improvement of the proposal presented in this paper. On the one hand, the number of features to consider grows exponentially with every expansion process. This degrades the performance of the methodology. To overcome this situation, an heuristic that restricts the number of features generated may be useful.

On the other hand, many of the related works employ external knowledge in order to generate new features. Thanks to the modular design of OWL, which greatly facilitates the interconnection among ontologies, it should not be so difficult to integrate information coming from external data sources into the dataset, once expressed in form of ontology. Our future work is also focused on developing an ontology that describes the environments of the ADL experiments and help us to interconnect the datasets with general purpose knowledge bases.

Finally, the methodology proposed in this paper is general enough to be easily applied to any other domain. The conversion of the dataset in form of ontology is actually the most difficult part, because the rest of the tasks are independent of the problem domain. Our next objective is to apply this methodology in the field of text mining.

Acknowledgments

This project has received funding from the Spanish government, by research project TIN2015-66524-P, and the REMIND Project, from the European Union's Horizon 2020 research and innovation programme, under the Marie Skłodowska-Curie grant agreement No 734355.

References

[1] Gutiérrez López de la Franca C, Hervás R, Johnson E, Mondéjar T, Bravo J. Extended body-angles algorithm to recognize activities within intelligent environments. J Ambient Intell Humaniz Comput 2017;8(4):531–49. http://dx.doi.org/10.1007/s12652-017-0463-y.
[2] Ferrández-Pastor FJ, Mora-Mora H, Sánchez-Romero JL, Nieto-Hidalgo M, García-Chamizo JM. Interpreting human activity from electrical consumption data using reconfigurable hardware and hidden markov models. J Ambient Intell Humaniz Comput 2017;8(4):469–83.
[3] Espinilla M, Medina J, Calzada A, Liu J, Martinez L, Nugent C. Optimizing the configuration of an heterogeneous architecture of sensors for activity recognition, using the extended belief rule-based inference methodology. Microprocess Microsyst 2017;52(Supplement C):381–90. http://dx.doi.org/10.1016/j.micpro.2016.10.007.
[4] Li C, Lin M, Yang L, Ding C. Integrating the enriched feature with machine learning algorithms for human movement and fall detection. J Supercomput 2014;67(3):854–65.
[5] Nugent C, Synnott J, Santanna A, Espinilla M, Cleland I, Banos O, Lundstrom J, Hallberg J, Calzada A. An initiative for the creation of open datasets within the pervasive healthcare. Proceedings - 10th EAI international conference on pervasive computing technologies for healthcare. 2016. p. 180–3.
[6] Chen L, Nugent C, Wang H. A knowledge-driven approach to activity recognition in smart homes. IEEE Trans Knowl Data Eng 2012;24(6):961–74.
[7] Khattak AM, Khan WA, Pervez Z, Iqbal F, Lee S. Towards a self adaptive system for social wellness. Sensors 2016;16(4). http://dx.doi.org/10.3390/s16040531.
[8] Chen L, Nugent C, Okeyo G. An ontology-based hybrid approach to activity modeling for smart homes. IEEE Trans Hum Mach Syst 2014;44(1):92–105. http://dx.doi.org/10.1109/THMS.2013.2293714.
[9] Rafferty J, Chen L, Nugent C, Liu J. Goal lifecycles and ontological models for intention based assistive living within smart environments. Comput Syst Sci Eng 2015;30(1):7–18.
[10] Cheng W, Kasneci G, Graepel T, Stern D, Herbrich R. Automated feature generation from structured knowledge. Proceedings of the 20th ACM international conference on information and knowledge management. ACM; 2011. p. 1395–404.
[11] Terziev Y. Feature generation using ontologies during induction of decision trees on linked data. ISWC PhD symposium. 2016.
[12] Paulheim H. Generating possible interpretations for statistics from linked open data. Semant Web 2012:560–74.
[13] Brown M, Hua G, Winder S. Discriminative learning of local image descriptors. IEEE Trans Pattern Anal Mach Intell 2011;33(1):43–57.
[14] Van Kasteren T, Noulas A, Englebienne G, Kröse B. Accurate activity recognition in a home setting. UbiComp 2008 - Proceedings of the 10th international conference on ubiquitous computing. 2008. p. 1–9.
[15] Singla G, Cook DJ, Schmitter-Edgecombe M. Tracking activities in complex settings using smart environment technologies. Int J Biosci Psychiatr Technol IJBSPT 2009;1(1):25.
[16] Ordónez FJ, de Toledo P, Sanchis A. Activity recognition using hybrid generative/discriminative models on home environments using binary sensors. Sensors 2013;13(5):5460–77.
[17] Yin J, Tian G, Feng Z, Li J. Human activity recognition based on multiple order temporal information. Comput Electr Eng 2014;40(5):1538–51. http://dx.doi.org/10.1016/j.compeleceng.2014.04.006.
[18] Baryannis G, Woznowski P, Antoniou G. Rule-based real-time adl recognition in a smart home environment. International symposium on rules and rule markup languages for the semantic web. Springer; 2016. p. 325–40.
[19] Bae I-H. An ontology-based approach to adl recognition in smart homes. Future Gener Comput Syst 2014;33:32–41.
[20] Noor MHM, Salcic Z, Kevin I, Wang K. Enhancing ontological reasoning with uncertainty handling for activity recognition. Knowl Based Syst 2016;114:47–60.
[21] Lehmann J, Auer S, Bühmann L, Tramp S. Class expression learning for ontology engineering. Web Semant 2011;9(1):71–81. http://dx.doi.org/10.1016/j.websem.2011.01.001.
[22] Aloulou H, Mokhtari M, Tiberghien T, Endelin R, Biswas J. Uncertainty handling in semantic reasoning for accurate context understanding. Knowl Based Syst 2015;77:16–28. http://dx.doi.org/10.1016/j.knosys.2014.12.025.
[23] Drummond N, Rector A, Stevens R, Moulton G, Horridge M, Wang H, Seidenberg J. Putting owl in order: Patterns for sequences in owl. OWLED. 2006.
[24] ODP. Owl list pattern. http://ontologydesignpatterns.org/wiki/Submissions:List; 2010. Accessed: 2017-05-18.
[25] Eibe F., Hall M., Witten I., Pal J.. The weka workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques 2016; 4.

Alberto G. Salguero received a B.Sc. in 2001 and M.Sc. in Computer Science from de University of Granada, Spain in 2004, and a Ph.D. in Computer Sciences from the University of Cádiz, Spain, in 2013. He is an Interim Professor of Computer Sciences in the University of Cádiz since 2010. His research interests are ontologies, information systems and the recognition of ADL.

Macarena Espinilla Estévez received the M.Sc. and Ph.D. degrees in Computer Sciences from the University of Jaén, Spain, in 2006 and 2009, respectively. She is currently an Associate Professor with the Department of Computer Science, University of Jaén. Her research interests include ambient assisted living, activity recognition, ambient intelligence, group decision making, decision support systems, intelligence systems, fuzzy logic and linguistic modeling.