Hindawi
Scientific Programming
Volume 2019, Article ID 2917294, 19 pages
https://doi.org/10.1155/2019/2917294

Research Article

Ontology-Based Framework for the Automatic Recognition of Activities of Daily Living Using Class Expression Learning Techniques

Alberto G. Salguero,[1] **Pablo Delatorre**,[1] **Javier Medina**,[2] **Macarena Espinilla**,[2] **and Antonio J. Tomeu**[1]

[1]*Computer Science Department, University of Cádiz, Cádiz 11519, Spain*
[2]*Computer Science Department, University of Jaén, Jaén 23009, Spain*

Correspondence should be addressed to Alberto G. Salguero; alberto.salguero@uca.es

Received 29 September 2018; Revised 31 January 2019; Accepted 25 February 2019; Published 1 April 2019

Academic Editor: Fabrizio Riguzzi

The miniaturization and price reduction of sensors have encouraged the proliferation of smart environments, in which multitudinous sensors detect and describe the activities carried out by inhabitants. In this context, the recognition of activities of daily living has represented one of the most developed research areas in recent years. Its objective is to determine what daily activity is developed by the inhabitants of a smart environment. In this field, many proposals have been presented in the literature, many of them being based on *ad hoc* ontologies to formalize logical rules, which hinders their reuse in other contexts. In this work, we propose the use of class expression learning (CEL), an ontology-based data mining technique, for the recognition of ADL. This technique is based on combining the entities in the ontology, trying to find the expressions that best describe those activities. As far as we know, it is the first time that this technique is applied to this problem. To evaluate the performance of CEL for the automatic recognition of activities, we have first developed a framework that is able to convert many of the available datasets to all the ontology models we have found in the literature for dealing with ADL. Two different CEL algorithms have been employed for the recognition of eighteen activities in two different datasets. Although all the available ontologies in the literature are focused on the description of the context of the activities, the results show that the sequence of the events produced by the sensors is more relevant for their automatic recognition, in general terms.

1. Introduction

The advancement of technology allows developing smaller and cheaper sensors. This facilitates the creation of smart environments, where many sensors capture the actions carried out by their inhabitants. The objective of these smart environments is to increase the safety of the inhabitants, to enhance the efficiency in the development of the activities, or simply, to improve the users experience [1]. Based on these trends, as well as the recent emergence and popularity of smart environments and pervasive computing, multiple approaches for the automatic recognition of activities of daily living (ADL) have been developed.

In this case, the problem consists of determining the activities that the inhabitants of the smart home are performing based on the information provided by the set of sensors. It is not a trivial task because (i) many of the activities involve the activation of multiple shared sensors, (ii) there are also activities in which there is no single sensor activation sequence, or (iii) there may be overlapped activities taking place at the same time [2].

The automatic recognition of such ADL is usually based on artificial intelligence tools and techniques. In this field, many proposals have been presented in the literature, many of them being based on logical rule systems. Among them, ontology-based approaches have provided very successful

results in both data-driven approaches (DDAs) [3, 4] and knowledge-driven approaches (KDAs) [5–7]. DDA usually relies on machine learning techniques in which a preexistent annotated dataset of user behaviors is required. A training process is carried out to build an activity model which is followed by a testing process that evaluates the generalization of the model in classifying unseen activities [4]. With KDA, an activity model is built through the incorporation of a rich prior domain knowledge gleaned from the application domain, using knowledge engineering and knowledge management techniques [1].

There is a wide variety in types of rules engines and general logic processors or "reasoners." The high formalization of ontologies coupled with description logic (DL) reasoners allows us to extract high-quality knowledge, which has not been given explicitly, through a process of automatic reasoning. However, there is no standard model for the description of ADL using ontologies. The lack of a standard model requires the creation of *ad hoc* ontologies for each particular case, whose resulting ontologies are difficult to reuse due to the incorporation of specific context-aware information and how the ADL are described in each work. In fact, all the ontologies available in the literature are mainly focused on describing the context in which activities happen. They usually include information about the location and the type of the sensors, as well as information about the inhabitant who were doing the activity. In some cases, there is even a distinction between complex activities and simpler actions. However, none of the available ontologies in the literature considers the order in which events occur. Aware of this situation, we proposed an ontology in a previous work specifically designed to represent the sequences of sensor events [3, 8]. It was, actually, the only information about activities considered in the ontology. No more information about the activities or the sensors was included.

In this paper, the ontology that we previously proposed is extended to include more information about the contexts in which the activities happen. The ontology has been developed around a core ontology in which the basic entities for describing ADL are defined. These entities constitute the minimum necessary concepts and relations to properly represent the ADL as sequences of events detected by the sensors in the smart environment. When necessary, the core ontology may be extended by importing other ontologies, which can be used to describe the type of the sensors, their locations, or the actions the ADL are composed of, for example. This modular approach reduces the amount of entities the reasoners have to deal with and improves the efficiency of the whole process.

Secondly, in this work, we propose and evaluate the use of class expression learning (CEL) for the recognition of ADL. This technique basically combines all the concepts and properties in the ontology, trying to find the expressions that best describe each of those activities. In the case of the OWL ontology, the CEL algorithms combine the concepts and properties using the DL operators. Regarding the experiment in this work, two different CEL algorithms have been employed for the recognition of eighteen activities in two well-known datasets. CEL is a DDA technique that, as far as we know, has been applied to this problem the first time.

The remainder of the paper is structured as follows: in Section 2, we introduce some notions about ontologies that are needed to understand our proposal. Some related works in the literature are also reviewed in this section. Section 3 introduces the CEL technique. In Section 4, we describe the general architecture of the framework and the extended version of the ontology we have developed. Section 5 describes the experiment that we have conducted to evaluate the performance of different ontologies and CEL algorithms for the recognition of the activities. In Section 6, we analyze and discuss the results obtained in the experiment. Finally, conclusions are presented in Section 7.

2. Background

In this section, some concepts related to ontologies are reviewed and some ontology-based approaches for data mining are introduced. Also, some related works are presented.

2.1. 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 [9, 10]. The ontologies are more complex than taxonomies because they not only consider the *type of* relations, but they also consider other relations, including *part of* or domain-specific relations [11].

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

The semantics of operators to combine 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 interpretation, and I is an interpretation function [12].

An ontology expresses what individuals, also called objects, belong to which concepts. Moreover, it is possible to declare properties to relate individuals, organizing them into a hierarchy of subproperties and providing domains and ranges for them. Usually, the domains of properties are concepts and ranges are either concepts or data types. A declared property can be defined as transitive, symmetric, functional, or the inverse of another property (R^-).

The main advantage of ontologies is that they codify knowledge and make it reusable by people, databases, and applications that need to share information [11, 13]. Due to this, the construction, the integration, and the evolution of ontologies have been critical for the Semantic Web [14–16] or Internet of the Things [17, 18]. 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 [19, 20], which is developed by the World Wide Web

Table 1: Semantics of OWL logical operators.

DL syntax	Semantics
$C_1 \sqcap C_2$	$(C_1 \sqcap C_2)^I = (C_1^I \cap C_2^I)$
$C_1 \sqcup C_2$	$(C_1 \sqcup C_2)^I = (C_1^I \cup C_2^I)$
$\neg C$	$(\neg C)^I = \Delta^I \setminus C^I$
$\exists R \cdot C$	$(\exists R \cdot C)^I = \{x \mid \exists y \cdot \langle x, y\rangle \in R^I \wedge y \in C^I\}$
$\forall R \cdot C$	$(\forall R \cdot C)^I = \{x \mid \forall y \cdot \langle x, y\rangle \in R^I \longrightarrow y \in C^I\}$
$\leq nR \cdot C$	$(\geq nR \cdot C)^I = \{x \mid \text{card}\{y \cdot \langle x, y\rangle \in R^I \wedge y \in C^I\} \leq n\}$
$\geq nR \cdot C$	$(\leq nR \cdot C)^I = \{x \mid \text{card}\{y \cdot \langle x, y\rangle \in R^I \wedge y \in C^I\} \geq n\}$

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 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.

The design of OWL is greatly influenced by DL, particularly in the formalism of semantics, the choice of language constructs, and the integration of data types and data values.

2.2. Related Works. In the literature, there are many proposals that employ ontologies for the recognition of ADL. In previous works, we also proposed an ontology for the representation of ADL [3, 8]. However, this ontology was focused solely on the identification of the events produced by the sensors, ignoring the rest of the information about the activities, such as the types of the sensors involved or the rooms where they were located. Actually, there were only three concepts defined in that ontology: *Activity*, *Event*, and *Sensor*.

Among all available proposals, two different main approaches are distinguished: KDA-based approaches and hybrid approaches. In KDA-based approaches, the ontologies usually contain rules to correctly identify different kinds of activities. Those rules are usually handcrafted and expressed in the form of DL axioms which can be natively processed by OWL reasoners. Some other proposals make use of external, custom rule systems to identify the corresponding ADL. As the logic behind ontologies is very rigid, other hybrid approaches employ ontological reasoning coupled with statistical reasoning in order to address the problem. Symbolic reasoning in these approaches is usually employed to refine the statistical inference. Table 2 summarizes the most important features of the ontologies revised in this section.

2.2.1. KDA-Based Approaches. SPHERE ADL [21] is an ontology that was specifically developed in the University of Bristol for representing their collected datasets in the context of the SPHERE (Sensor Platform for HEalthcare in Residential Environment) project. The SPHERE architecture attempts to combine different sensing technologies to provide a generic platform for ADL recognition. The ontology is written in OBO format and contains one hundred sixty-five predefined activities. ADL are organized hierarchically, grouping them into categories such as *Health condition*, *Social interaction*, or *Atomic home activities*. *Atomic home activities* capture the low-level activities or simple actions which form the basic building blocks for other activities. These include actions such as *open door* or *cupboard close*. Since each of the *Atomic home activities* is associated with a unique physical sensor, we may treat them as an abstraction of the sensor data stream. It also incorporates other types of entities that are specific for the datasets, such as the person who is performing the activity and his or her posture. All defined properties are flat, i.e., without any special feature (inverse, functional, etc.), because the identification of ADL is done in an external, *ad hoc* rules system [28] implemented in Jess, a Java rule-based system.

The ontology proposed in [7] is oriented towards the development of an ADL 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. It is one of the few ontologies in which properties are enriched with their ranges and domains, and some of them have been declared as functional or inverse properties. This facilitates the automatic discovering of information by reasoners. Unfortunately, the ontology is not publicly available, so it is not possible to study it in depth.

Noor et al. [23] proposed one of the few ontologies that is able to deal with sensor events. They investigated the fusion of wearable devices and ambient sensors for recognizing ADL. They propose a set of rules that must be fulfilled by the activities in order to be considered of one type or another. The developed ontology also differentiates among simple actions and composite activities. An activity such as "preparing meal" could be defined as a sequence of actions such as "taking pot," "opening microwave oven door," and "closing microwave oven door." Furthermore, other features of the context are considered when describing activities, such as the location of the resident or his or her posture. They introduced a concept called *Interval* to model a context by specifying the associated sensor states. The intervals may be related to their preceding intervals by using a transitive property. However, it is not possible to specify the order in which sensor events occur in the intervals.

SOCOM [27] is a generic multisensor-oriented ontology for context-aware computing. It has not been specifically developed for ADL recognition, but it incorporates some concepts that may be used for this task. It is focused on sensors description, providing a comprehensive, generic sensor category classification and sensor properties frequently used in different context-aware systems. SensorML [29] is used as the base for describing metainformation about sensors and their capabilities. Sensors are first

Table 2: Features of existing ADL ontologies.

	Salguero et al. [3]	SPHERE [21]	COSAR [22]	Noor [23]	Chen and Nugent [24]	Bae [7]	Hois [25]	ODI [26]	SOCOM [27]
Approach	DDA	KDA	Hybrid	KDA	KDA/hybrid	KDA	KDA	KDA	KDA
Format	OWL	OBO	OWL	OWL	OWL	OWL	OWL	OWL	OWL
Sensor events	✓	✓*	✓	✓			✓		✓
Ordered events	✓								
Overlapped activities		✓	✓						
Multiagent			✓			✓			
Locations		✓	✓	✓	✓	✓	✓	✓	✓
Sensors hierarchy		✓*	✓		✓	✓			✓
Composite activities		✓		✓	✓				
Posture		✓	✓	✓					
Enriched properties	✓			✓		✓			
Available	✓	✓	✓	✓					

*Although sensor events are not explicitly defined, we can consider atomic activities as such in this model since each atomic activity is directly related to the activation or deactivation of a single sensor.

categorized into physical sensors and logical sensors. A physical sensor is a physical device that captures or detects the environmental elements in real world, such as GPS or flow sensors. A logical sensor is a data source that indirectly interacts with the physical layer, and it often has no computational ability, such as calendars or battery status. Logical sensors may combine different physical sensors and other logical sensors to form a higher-level context, which may be seen as the sensed phenomenon. In this model, the *Activity* concept is a specialization of the *Entity* concept, as well as the concepts *Person* or *Event*. Sensors are deployed at entities, and entities characterize contexts, providing their location and temporal and quality information.

2.2.2. Hybrid Approaches. Hybrid approaches perform probabilistic reasoning or employ external rules systems, different than DL reasoning. The work proposed in [5] is a clear example. They defined an OWL ontology to formally model the smart home environments and the semantics of activities in two well-known datasets. Artifacts in the smart homes are organized in a hierarchy, *Stove* being a subclass of *Cooking instruments*, for example. The ontology also models sensors and the operation that they detect; e.g., a power sensor attached to the electric stove detects the operation turning on the stove. In turn, this operation is a subclass of *Cooking instrument*. They translate their ontological model into a Markov logic network (MLN) and perform probabilistic reasoning to refine candidate activity instances. Correlations among sensor events and activities are computed to infer, for each event, the most probable activity generating it. A set of *ad hoc* rules are defined in order to derive necessary conditions about the sensor events that must occur during the execution of a specific activity, such as the following: "since the stove is the only cooking instrument in the home, and a sensor is available that detects the usage of the stove, then each instance of preparing hot meal executed in the home must necessarily generate an event from that sensor." Rules may also consider other necessary conditions regarding time and location. This includes constraints on the duration of the activity instance and dependencies between activity and location. As can be assumed, these rules are very specific and they only have sense in the context of particular datasets. They are plenty of references to artifacts and conditions that are only available in those datasets. The same ontology is also used by Gayathri et al. to propose another MLN-based hybrid approach for ADL recognition [30]. The sensors data stream is first described in form of ontology. The information in the ontology is then automatically transformed to first-order logic axioms. An open source tool for MLN is then used to perform weight learning and carry out probabilistic reasoning over first-order logic axioms. The ontology is actually an extension of the COSAR ontology [22], which was originally intended to model context data and human activities. The extension mainly regarded the definition of a few classes for activities and artifacts that were not considered in the COSAR ontology. A limitation of this ontology is that the temporal information and duration of the activities are not effectively modeled.

Chen et al. [6] developed an ontology for the modeling of ADL that has been employed for both KDA and hybrid approaches [31]. They introduced the concept of context modeling to relate temporal and spatial information to activities performed by inhabitants. Spatial contexts include the location and the surrounding entities, such as rooms, furniture, and appliances. The event contexts are used to record the state changes of sensors, while the environmental contexts contain information about temperature humidity and other weather conditions. Temporal context indicates the time and/or duration of activities. The ontology does not explicitly record all the sensors events. Instead, the sequences of sensor activations are aggregated to generate primitive activities performed by the user at a specific time point. Composites ADL are then defined combining those primitive activities, in the same way as simple actions and complex activities are proposed by Noor et al. [23]. Multiple activity classes are defined to represent the activities in some

datasets, and they are structured in a hierarchical tree. Classes near the root of the tree describe generic activities, while classes near the leaves include many more properties and represent more specific activities.

Hois [25] proposed a set of ontologies for the description of ambient-assisted living environments. Those ontologies allow the description of intelligent environments from different points of view, including information related to architectural building elements, such as walls or windows; functional information of room types and sensors; user actions, such as cooking or taking a shower; types of furniture or devices inside the smart environment; and requirements and constraints of the assisted living system, such as temperature regulations. The ontologies have been developed following a modular approach, which allows the selection of application-specific ontologies as necessary. However, the ontologies are not specially focused on the recognition of ADL but towards the development of applications for the automation of smart environments, such as control of lighting, air conditioning, or access restriction. The proposed application, for example, monitors the temperature sensors and evaluates the class descriptions from time to time. When abnormal situations are detected (class restrictions are not satisfied), actions to improve the comfort of the inhabitants are suggested. Unfortunately, the ontologies are not currently available, so it is not possible to analyze them with more detail.

Previous works on the Open Data Initiative (ODI) framework have involved the development of *homeML*, a standard vocabulary for describing experiments in smart environments. However, they have recently proposed the use of an ontological model for replacing the *homeML* vocabulary [26]. The ontology proposed is actually a model for describing experiment metadata. The sensor data stream is stored in an external file using the eXtensible Event Stream (XES) standard for event data storage. The ontology is used to describe the entities involved in the experiment, such as the rooms, the inhabitants, or the activities being performed. The ontology distinguishes between single participant activities and multiparticipant activities, as well as between fixed location and mobile devices. Typical queries the ontology might answer include: Which ADLs have been considered within the experiments? Which experiments involve recording events in a kitchen?

As we have discussed in this section, there are many ontologies related to the recognition of ADL. However, the main lack is that just few of them are really available. In addition, in many of the available ontologies in the literature, the stream of sensor events is aggregated into higher-level actions, which hinger the automatic recognition of patterns. In some other cases, the ontologies proposed by other authors are ontologies created *ad hoc* and contain specific information about the particular dataset that is being used, so they are difficult to be reused. Usually, most of these ontologies emphasize the description of the specific context in which the ADL are developed rather than the description of the activities themselves. For example, they contain classes and properties to describe and classify locations and sensors as well as information about the people who are performing them. Another important problem is that none of the ontologies proposed by other authors consider the order relation among the events produced in the sensors. The events are only related to the time interval in which they occur, but there is no order relationship among the events in the interval. The ontology we have proposed in previous works [3] employed the opposite approach, where the description of activities as sequences of events produced by the sensors were considered, ignoring the contextual information. For this reason, we propose in this work a generic ontology that allows the description of ADL as sequences of sensors events as well as including other kinds of contextual information of activities, such as their location or the posture of the inhabitants.

3. Class Expression Learning

A reasoner is an application that is able to infer logical consequences from a set of asserted facts or axioms. DL reasoning algorithms are often used in practice to compute a classification of a knowledge base $\mathcal{K}$, that is, to determine whether $\mathcal{K} \models A \sqsubseteq B$ for each pair of atomic concepts A and B occurring in $\mathcal{K}$ [32]. Reasoners may also be used to determine whether or not an ontology is consistent, to perform the classification of the individuals, and much more. However, the reasoners are not designed to suggest the existence of new atomic concepts nor more complex class descriptions. For this reason, researchers have investigated the use of Inductive Logic Programming (ILP) for learning logical theories since the mid 80s [33]. Among them, Maedche and Staab [34] coined the concept of "ontology learning" almost twenty years ago in the context of the automatic generation of ontologies. The existing approaches for ontology learning can be roughly classified into the following [35]:

(i) *Ontology Learning from Text* approaches mostly focus on the automatic generation of taxonomies from texts, employing lexicosyntactic patterns for hyponymy detection [36] or named-entity classification [37], for example.

(ii) *Linked Data Mining* approaches try to find meaningful patterns in RDF graphs [38].

(iii) *Concept Learning in Description Logics and OWL* approaches are usually based on Inductive Logic Programming methods, and their goal is to find new relevant class descriptions by means of supervised machine learning algorithms [35, 39].

(iv) *Crowdsourcing* approaches combine the power of machine-based approaches and humans to construct more complete and accurate taxonomies [40].

CEL technique falls into the category of concept learning in description logics and OWL. Its objective is to determine new class descriptions for concepts that may be used to classify individuals in an ontology according to some criteria. More formally, let a concept name $Target$, a knowledge base $\mathcal{K}$ with sufficient number of named individuals N_I (not containing $Target$), and sets E^+ and E^- be given, where

(i) $E^+ \subseteq N_I$ is the set of individuals in $\mathcal{K}$ such that $x \in E^+ \Longrightarrow x \in E^I$ in every interpretation I of the knowledge base

(ii) $E^- \subseteq N_I$ is the set of individuals in $\mathcal{K}$ such that $x \in E^- \Longrightarrow x \notin E^I$ for at least one interpretation I of the knowledge base

The learning problem is to find a concept C such that $Target$ does not occur in $\mathcal{K}$, and for $\mathcal{K}' = \mathcal{K} \cup \{\text{Target} \equiv C\}$, we have $\mathcal{K}' \models E^+$ and $\mathcal{K} \not\models E^-$, where $\mathcal{K} \models S$ means that every element in S follows from $\mathcal{K}$ and $\mathcal{K} \not\models S$ means that no element in S follows from $\mathcal{K}$ [41].

As it can be seen, the CEL problem is actually a kind of supervised learning problem. However, the features of the instances that are used in classic supervised learning algorithms are initially unknown for this problem. They are dynamically generated as the CEL algorithm moves along the search space S, composed of all the possible concepts in $\mathcal{K}$. A key point of many ILP approaches are the refinement operators, which are used to traverse the search space [41].

Definition 1 (refinement operator). A *quasi-ordering* is a reflexive and transitive relation. In a quasi-ordered space $(S, \preccurlyeq)$, a *downward (upward)* refinement operator ρ is a mapping from S to 2^S such that for any $C \in S$, we have that $C' \in \rho(C)$ implies $C' \preccurlyeq C (C \preccurlyeq C')$. C' is called a specialization (*generalisation*) of C.

Definition 2 (refinement chain). A refinement chain of refinement operator ρ of length n from a concept C to a concept D is a finite sequence $C_0, C_1, \ldots, C_n$ of concepts, such that $C = C_0, C_1 \in \rho(C_0)$, $C_2 \in \rho(C_1), \ldots, C_n \in \rho(C_{n-1})$, $D = C_n$. This refinement chain goes through E iff there is an $i\,(1 \le i \le n)$ such that $E = C_i$. We say that D can be reached from C by ρ if there exists a refinement chain from C to D.

Definition 3 (downward and upward cover). A concept C is a downward cover of a concept D iff $C \sqsubset D$ and there does not exist a concept E with $C \sqsubset E \sqsubset D$. A concept C is an upward cover of a concept D iff $D \sqsubset C$ and there does not exist a concept E with $D \sqsubset E \sqsubset C$.

Theoretical investigations on refinement operators have identified desirable properties for them to have, which impact their performance. These properties thus provide general guidelines for the definition of suitable operators [41].

Definition 4 (properties of DL refinement operators). A refinement operator is called as follows:

(i) *finite* iff $\rho(C)$ is finite for all concepts C.

(ii) *redundant* iff there exists a refinement chain from a concept C to a concept D, which does not go through some concept E, and a refinement chain from C to a concept weakly equal to D, which does go through E.

(iii) *proper* iff for all concepts C and D, $D \in \rho(C)$ implies $C \equiv D$.

(iv) *ideal* iff it is finite, complete (Definition 5), and proper.

Definition 5 (downward refinement operator). A downward refinement operator is called

(i) *complete* iff for all concepts C and D, with $C \sqsubset D$, we can reach a concept E with $E \equiv C$ from D by ρ.

(ii) *weakly complete* iff for all concepts $C \sqsubset \top$, we can reach a concept E with $E \equiv C$ from $\top$ by ρ.

(iii) *minimal* iff for all C, $\rho(C)$ contains only downward covers and all its elements are incomparable with respect to $\sqsubseteq$.

The corresponding upward refinement operator is defined dually.

In addition to this operator, it is also necessary to establish a search strategy in S that maximizes the searched area and avoid the analysis of already visited areas. In literature, we can find several proposed search strategies [42, 43], which are usually based on graph exploration algorithms. There are also some of them that are based on computational intelligence, like genetic algorithms [41], where the refinement operator consists of the combination of existing classes in the knowledge base. Badea et al. invented a refinement operator for $\mathcal{ALER}$ and proposed to solve the CEL problem by using a top-down approach [44]. The YINYANG tool combines both techniques for constructing ontologies in a semiautomatic fashion [45]. However, those algorithms tend to produce very long and hard-to-understand class expressions. The algorithms implemented in the DL-Learner tool try to overcome this problem by biasing them towards the generation of shorter class expressions [39]. DL-FOIL, which is based on a mixture of upward and downward refinement of class expressions, employs a similar approach [46]. Some extension of the latter have been proposed for dealing with fuzzy extensions of DL [47] or to avoid suboptimal solutions due to the kind of refinement operators being used [48]. Another approach to concept learning is based on *bisimulation* [49, 50]. Instead of trying to specialize or generalize solutions, the *bisimulation* method exploits a set of predefined *selectors*, i.e., tests that are used to partition the set of individuals.

Algorithm 1 represents a very basic implementation of a CEL algorithm. First, the algorithm gets the current class description that best fit C^+ and C^-. Actually, the heuristic for selecting the best class descriptions depends on the specific algorithm. They may consider many factors, such as the accuracy for classifying positive and negative instances or the length of the class expression. The best class description found is then combined with all of the other class descriptions in the knowledge base using a selected refinement operator. The process is restarted until the stopping condition is met. In this case, the algorithm stops when a given number of class descriptions are generated.

Implementations for the OCEL (OWL class expression learner), ELTL (EL tree learner), and CELOE (class expression learning for ontology engineering) can be found in

[51]. OCEL is the standard learning algorithm employed in the DL-Learner tool. ELTL is an algorithm optimized for learning $\mathcal{EL}$ concepts, and CELOE is an optimized version of OCEL.

Example 1. As an example, let us suppose the existence of a family ontology O, having a sufficient number of individuals, where the concepts *Male*, *Female*, *Parent*, and *Child* and the property *hasChild* are defined conveniently. Let us suppose we want to automatically find a description for a new concept called *Father*.

(1) First, the user defines the $Father^+$ set by selecting those individuals in O that should be classified as *Father*. The $Father^-$ set contains individuals that should be classified as $\neg Father$. Obviously, the *Father* concept should not be part of the ontology at this point of the process.

(2) Using a refinement operator ρ, the search space S is traveled. During this travel, a set of class descriptions $Z \subseteq S$ is generated, where the classes and properties in O are combined using the DL operators. Following the example of the family ontology, the class descriptions $Z = \{\neg Male, Male \sqcap Female, \exists\ hasChild \cdot \top, \forall\ hasChild \cdot (\neg (Parent \sqcup Child))\}$ may eventually be generated.

(3) For each class description $z_i \in Z$, the sets z_i^I and $(\neg z_i)^I$ are calculated. The process is repeated again until some z_i is found such that $z_i^I = Father^+$ and $(\neg z_i)^I = Father^-$, after a given period of time has passed or a given number of class expressions have been generated. CEL algorithms usually give the class descriptions that best approximate the *Father* concept as result in the latter cases.

If the process runs for enough time, the CEL algorithm will eventually find a class description $z_i = Male \sqcap \exists\ hasChild \cdot \top$ in the second step. Assuming the individuals in O are correctly annotated, $z_i^I = Father^+$ and $(\neg z_i)^I = Father^-$, so the process will stop and z_i will be proposed as a solution.

4. A Framework for the Mining of ADL

In this section, we describe our generic ontology-based framework for the mining of activities of daily living (ADL) (OMA), which is composed of a set of extensible applications and an ontology for the description of the activities (all the applications and ontologies have been published under the GPL open source license at https://sourceforge.net/projects/adl-mining-framework/). First, in this section, we describe the architecture of the framework and how the applications collaborate to produce the description of the activities in the form of an ontology. Next, we detail the core of the ontology, which contains the minimum entities necessary for the description of any activity in any dataset. Finally, we introduce some of the extensions we have implemented to add information about the context to the activities.

4.1. Framework Architecture. The architecture of the framework that we present in this work consists of three independent applications. The first two applications are located in the preprocessing stage, whose objective is to describe in a common data model the information contained in any dataset. The third application is responsible for transforming the information into an ontology, once expressed in the common data model.

We have chosen the eXtensible Event Stream (XES) standard as the common data model, which has been explicitly designed for the event data representation. In fact, XES has already been proposed by other authors as a common data model prior to the transformation of data into an ontology [26]. Since it is a standard data model for the representation of streams of sensors events, there are many applications available for working with this data model and to import data in other formats. Unfortunately, most of the datasets for the automatic recognition of ADL are expressed in formats that have been created *ad hoc* by their developers and there are no importers available. Therefore, the first step in the architecture is the translation of the information in the datasets to the XES format. This is the first task of the developed applications, which we call "XES Converter." This application converts to the XES format twelve of the sixteen annotated datasets from the CASAS repository (http://casas.wsu.edu/datasets/), as well as other well-known datasets, such as in [52, 53].

The XES standard is basically an XML grammar that provides common terms for describing streams of events. XES defines a stream of sensors events as a set of *traces*, which are themselves sequences of *events* (see Figure 1). The *log*, *traces*, and *events* may all contain one or more key-value attributes.

The XES model allows to easily describe the stream of events generated by a set of sensors. However, we have to note that datasets are usually designed to employ supervised learning techniques, so they also include annotations about the activities that were being carried out at each instant. To handle this situation, we have considered the following alternatives:

(1) The format XES has been designed to be easily extensible. Extensions are generally used to include new attributes in standard entities. One solution to describe both the stream of events and the stream of activities at the same time in the standard XES model consists in creating an extension that incorporate an attribute in the *traces* element to distinguish between the *traces* that represent streams of events and *traces* that represents the stream of activities. This solution has the advantage that all the information in the dataset is stored in a single file in the XES format.

(2) Another possibility, which does not require the elaboration of an extension, is to generate two files in the XES format. The first represents the stream of events as a single trace. The second represents the stream of activities, also represented as a single trace in XES. As detailed in Figure 2, this has been the chosen option. This solution allows us to visualize

Require: N_C is the set of named concepts in the knowledge base. n is the maximum number of class description the algorithm generates in the search process. α is a constant float value that indicates the importance of negative samples classification accuracy.

(1) **function** CEL (N_C, C^+, C^-, n)
(2) **while** $|N_C| < n$ **do**
(3) $b \longleftarrow$ BEST-DESCRIPTION (N_C, C^+, C^-)
(4) $N'_C \longleftarrow \varnothing$
(5) **for all** $c_i \in N_C$ **do**
(6) $d \longleftarrow \rho(b, c_i)$
(7) **if** valid(d) **then**
(8) $N'_C \longleftarrow N'_C \cup d$
(9) $N_C \longleftarrow N_C \cup N'_C$
(10) **return** b
(11) **function** BEST-DESCRIPTION (N_C, C^+, C^-)
(12) $v \longleftarrow -\infty$
(13) $c \longleftarrow c_0$
(14) **for all** $c_i \in N_C$ **do**
(15) $v_{\text{pos}} \longleftarrow |c_i^I \cap C^{+I}| / |C^{+I}|$
(16) $v_{\text{neg}} \longleftarrow |(\neg c_i)^I \cap C^{-I}| / |C^{-I}|$
(17) **if** $v_{\text{pos}} - \alpha \cdot v_{\text{neg}} > v$ **then**
(18) $v \longleftarrow v_{\text{pos}} - \alpha \cdot v_{\text{neg}}$
(19) $c \longleftarrow c_i$
(20) **return** c

Algorithm 1: CEL algorithm.

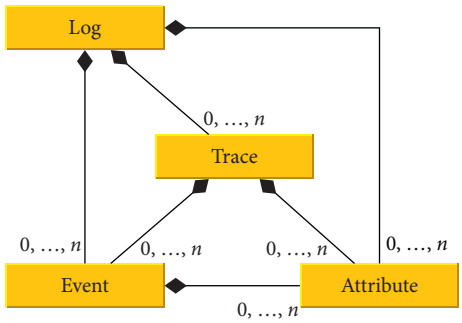

Figure 1: Entities in the XES basic model.

and analyze the information in all the available tools for the XES format.

ADL recognition approaches are usually based on the division of the stream of events into segments, also called windows or timeslots. Three main segmentation approaches can be found in the literature [54]:

(i) *Based on Activity.* This popular segmentation approach, also called explicit, is usually adopted by offline proposals. The stream of events is divided into segments coinciding with the starting and ending point of time for each activity.

(ii) *Based on Time.* The stream of events is divided into segments of a given duration. The main problem of this approach is to identify the optimal length of the segments.

(iii) *Based on Events.* Another approach consists in dividing the stream of events based on the number of sensor events. The main problem with this approach is to mix in the same timeslot sensor events that correspond to different activities.

(iv) *Others.* Other works include *ad hoc* segmentation approaches.

The "Segmenter" application is responsible for generating the final file in XES format, according to the temporal segmentation specified by the user. In this case, the resulting file contains one *trace* for each of the temporal windows that results from the segmentation process. Each *trace* contains the list of events produced within a unique temporal window. Since there are many methods to segment the stream of events, we decided to use the Plugin Framework for Java (PF4J) (http://www.pf4j.org) to develop this application. The PF4J library allows us to incorporate new segmentation approaches through plugins, without having to modify or recompile the application. Currently, four plugins have been developed, which correspond to (i) the three basic segmentation approaches described above and (ii) a custom and more elaborate type of segmentation which uses statistical criteria to develop time-based segmentation [54].

Once the dataset has been expressed in the XES format and the activities have been segmented, the dataset has to be translated into an ontology. The application "Ontology generator" is responsible for performing this operation. However, since the framework has been developed in a generic way, this application is not modeled under a previous specific ontology. Instead, we have used the PF4J framework again to delegate the actual construction of the ontologies to the plugins that can be added to the application. Currently, five different plugins have been developed to transform the dataset into four different ontologies available in the literature, apart from the one proposed in this work.

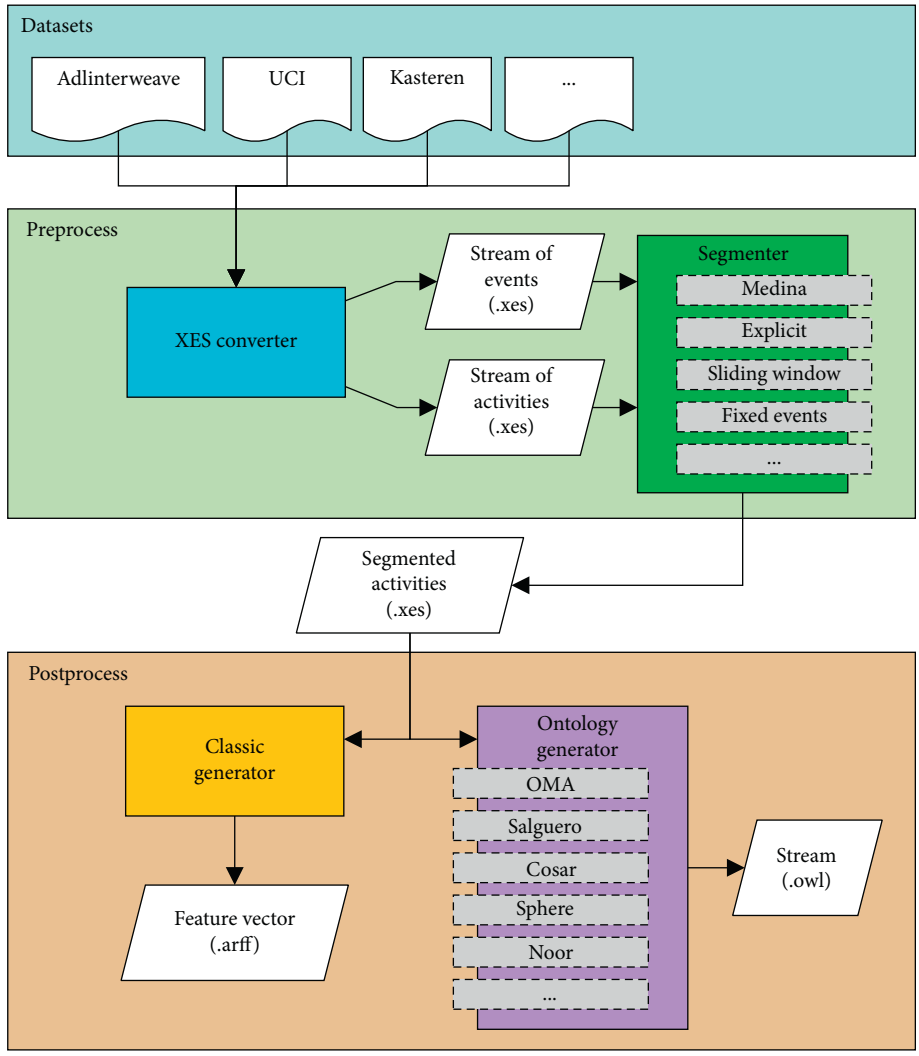

Figure 2: General architecture of the framework.

4.2. Core Ontology. As we discussed in Section 2, most of the available ontologies have been designed to be as expressive as possible, defining a huge number of classes and properties. The ontology proposed in [28], for example, contains one hundred sixty-five predefined activities and thirty-three different types of predefined actions or "atomic activities." This makes the search space to grow exponentially, hindering the data mining techniques. The core of the proposed ontology is only composed of the entities that are strictly necessary to describe the activities as sequences of events produced by the sensors. In fact, only three basic disjoint concepts are defined in this ontology, as shown in Figure 3: *Activity*, *Sensor*, and *Event*.

When performing an activity, it is necessary in many cases to carry out a temporal segmentation of the dataset, which usually consist of splitting the annotated activities into multiple smaller intervals. The concept *Activity* represents a single time interval of an activity in the dataset, during which this specific activity is being performed. Each interval must be created as an individual of this concept. It is important to note that there is no distinction among different types of activities at the core ontology. Concepts such as *answer the phone* or *dinner* are not defined here. If necessary, these concepts may be imported later from other ontologies. This allows us to reuse the core ontology with any dataset.

During the activities, the state of the sensors changes according to the actions of the inhabitants. The concept *Event* represents any situation or alteration reported by a sensor (such as its deactivation) which, in turn, is represented by the concept *Sensor*. As with the *Activity*, no particular types of events or sensors are defined at the core ontology.

Our proposal for representing activities is based on a list structure. The property *hasNextEvent* establish the order of the events in the activities. Since it has been defined as a functional property, just one event follows another event. The inverse property is also functional, forcing an event to be directly preceded by a unique event. *hasNextEvent* is a subproperty of the transitive property *isFollowedBy*. *hasNextEvent* is used to express that an event immediately follows another event. There is no other event between them.

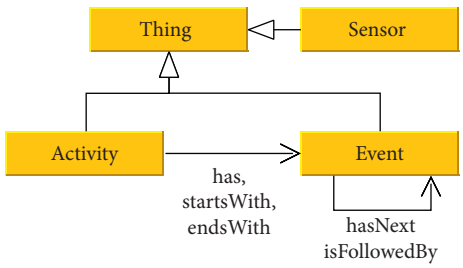

Figure 3: Entities in the kernel of the OMA ontology.

For example, *A hasNextEvent B* means *A* has *B* as the next event in the list or, in other words, event *A* is followed by event *B* (*A isFollowedByEvent B*). If another event *C* appears after event *B*, then event *A* is also followed by event *C* (*A isFollowedByEvent C*), but event *C* is not the next event of *A* in the list (*not A hasNext C*).

The property *hasEvent* relates an instance of an activity to all the events that occur during it. Due to open world assumption in OWL, reasoners cannot automatically infer the first and the last events of the activity. Therefore, it is necessary to annotate these individuals by using the properties *startsWithEvent* and *endsWithEvent*, which have been defined as subproperties of *hasEvent*. However, the simple annotation of these individuals in the ontology does not prevent having other elements before the first item of the list or after the last one. To avoid these situations, the *Activity* concept has been defined as a subclass of the class descriptions (1) and (2). Class description (1) describes the *Activity* concept as a list in which the first item is not preceded by any other item. Note that it is not necessary to explicitly declare in the ontology a property to represent the precedence relation since it can be expressed using the inverse of the property *isFollowedByEvent*. Analogously, class expression (2) prevents an activity from having an item after the last one in the list:

$$\neg\left(\exists\ startsWithEvent \cdot \left(\exists\ isFollowedByEvent^{-} \cdot \top\right)\right), \tag{1}$$

$$\neg\left(\exists\ endsWithEvent \cdot \left(\exists\ isFollowedByEvent \cdot \top\right)\right). \tag{2}$$

In OWL, the same individual could be referred 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 this reason, a functional property *hasID* has been defined to assign a unique code to all of the individuals in the ontology. This is much more efficient than asserting that all individuals are different.

The core of the OMA ontology has the exact same DL expressivity as the Salguero ontology ($\mathcal{SRIF}(\mathcal{D})$), being decidable in exponential time, as well as the COSAR ($\mathcal{ALCRIQ}(\mathcal{D})$) and Noor ($\mathcal{SR}(\mathcal{D})$) ontologies. Satisfiability of the Sphere ontology ($\mathcal{ALCI}$) is PSpace-complete. It is worth noting that we have included all the events and the activities of the datasets in the experiment because we needed all the positives and the negative examples. In a real-world application, just the last events are necessary to decide which activity has been performed by the inhabitant. Furthermore, the final application does not necessarily have to use ontologies nor reasoning mechanisms once the best descriptions of the activities have been found. It can be implemented using a simpler rule system, for example.

All these entities and relationships are automatically created by the plugin developed for the "Ontology generator" application. The concepts and relationships described in this subsection are defined in the file *kernel.owl* (Figure 4). However, the information in the dataset is not added to this file. Instead, the "Ontology generator" application generates an ontology in a separate file (*stream.owl*) which incorporates the entities defined in the file *kernel.owl* through the OWL import mechanism, without modifying or redefining the imported concepts. So, basically, the file *stream.owl* only contains individuals, which represent the events produced by the sensors and the instances of the activities collected in the dataset.

Apart from the information described in this subsection, the datasets usually contain other contextual information which can be extracted and processed automatically, such as the type of the sensor that generated each event or the exact value of flow sensors at a given instant. This kind of information can also be automatically generated by the plugin, which automatically includes the corresponding *import* directives. However, there is unstructured information about the dataset that have to be entered manually. This information includes, for example, the different rooms in which the smart environment is divided or the location of the sensors. Instead of modifying the file *stream.owl*, which is generated automatically by the "Ontology generator", it is highly recommended (i) to manually create a new file (called *ontology.owl* in Figure 4) to include this specific information about the dataset and (ii) to import the file *stream.owl* and all the necessary extensions from that file. This allows us to easily reuse the file containing specific information about the dataset in multiple experiments because *stream.owl*, the only file that changes among them, is generated automatically.

4.3. Extensions of the Core Ontology. Most of the ontologies available in the literature incorporate concepts and properties to describe ADL from different points of view [6, 28]. However, for efficiency reasons, the core ontology that we propose only contains the minimum entities necessary to represent the activities as sequences of events produced by the sensors, as previously described. In this way, the rest of the information about the activities must be added by importing other ontologies.

Following, we describe the ontologies that we have proposed as extensions of the core ontology. These ontologies act as modules that extend the entities in the core ontology. In addition, they can serve as a basis for the definition of other ontologies with much more specific information.

4.3.1. Sensor Ontology. This ontology describes the main types of sensors that can be found in the literature [55, 56],

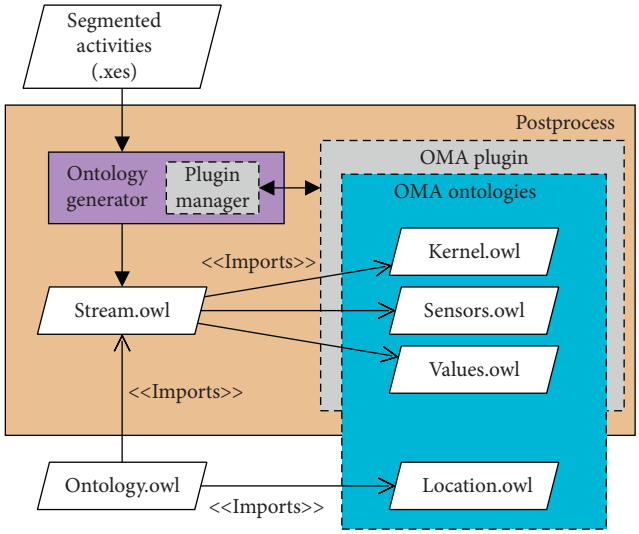

Figure 4: OMA plugin architecture.

such as motion detectors, contact sensor, and door sensors. All these concepts have been defined to be disjoint. The OMA plugin creates an individual of the corresponding category of sensors for each of the sensors in the smart environment. These individuals are related to the events they produce through the functional property *producedBySensor*.

4.3.2. Location Ontology. This ontology allows us to specify the location of the sensors. As the sensors' ontology, the location ontology defines the types of rooms that are usually found in ADL datasets, such as the kitchen, the bathroom, or the bedroom. All these concepts have been defined as disjoint to facilitate reasoners to distinguish between events produced at different locations. Unless stated, a reasoner may assume that the kitchen is the same room as the bedroom due to the open world assumption. Each of the rooms of the smart environment needs to be represented as an individual of the corresponding concept. Consequently, if the smart home has two bedrooms, two different individuals of the concept *Bedroom* will be necessary. The property *locatedAt* defines this fact in the ontology, allowing to relate a sensor to its location.

4.3.3. Values Ontology. This ontology is used to deal with the specific states of the sensors when the events occur. It contains a data property called *hasValue* that associates a literal value (integer, Boolean, string, etc.) with an event. Since most of the sensors that we can find in datasets are binary, two helper disjoint concepts have been defined in the ontology to represent this type of values: *Activated* and *Deactivated*. The concept $Activated \equiv hasValue(true)$ represents an activation of the sensor. For efficiency reasons, it is important to restrict the cardinality of the property *hasValue* for this concept ($Activated \sqsubseteq = 1\ hasValue.xsd:Boolean$) (for the sake of simplicity, we have abused the DL notation here: the operator = has been used to express the combination of both $\leq$ and $\geq$ cardinality restriction operators). Only two seconds are needed by the reasoner to evaluate all the individuals in the datasets of the experiment when the cardinality restriction is applied. Four minutes are required when it is not. The concept *Deactivated* is defined analogously. Note that the cardinality restriction in this module changes the DL expressivity of the OMA ontology ($\mathcal{SRIQ}(\mathcal{D})$), still being decidable in exponential time.

4.3.4. Time Ontology. This ontology allows to record the date and time at which an event occurred by defining the data type property *ocurretAt*. It also incorporates two analogous data type properties to record the starting and the ending instants of the activities.

4.3.5. Actor Ontology. In some of the ontologies in the literature, the inhabitant who carries out the activities is also considered. For this purpose, the *performedBy* property has been defined in this ontology, which relates the activity to the actor or actors that perform it, represented by the concept *Actor*.

4.3.6. Action Ontology. Complex ADL are often specified through sequences of simpler actions [23, 24, 30], such as heat water or take medicine. For this, the hierarchy of properties of the core ontology has been duplicated in this ontology, modifying their domains and ranges to relate activities and actions. These properties allow us to establish the order among actions in the activities. The duplicity of properties is for efficiency reasons. It could have been avoided if the domains and ranges of the properties of the core ontology had not been established. However, this would have had several negative effects: on the one hand, the inference capacity of the reasoners would have been reduced since they could not determine the type of individuals related by these properties; on the other hand, it would have been possible to define sequences composed of both simple actions and events, which makes more difficult the patterns recognition when applying data mining techniques. In addition, it is easier to detect errors and inconsistencies when the types of individuals the properties may relate are known.

5. Experiment

In order to evaluate the quality of the framework proposed in this work and the adequacy of CEL for activity recognition, an experiment has been carried out. The objective is to determine whether a particular ADL has been performed based on the sensors that have been fired during a specific period of time. The activity classification is binary. For each activity, the positive instances represent the individuals of the activity being recognized, while the negative instances represent the rest of the activities. CEL algorithms have to find a class description that describes all positive instances, but it does not include the negative ones. The positive and negative activities are annotated in the datasets. The application that is responsible for converting the datasets to the different ontologies also generates two lists with the names

of the individuals in the ontology that represent the positive and the negative instances.

For evaluation purposes, we have used an implicit segmentation, where the exact instants the activities start are not considered. Instead a time-based segmentation with a temporal window based on statistical criteria has been employed [54]. It provides a more challenging setting than the explicit segmentation, and it is closer to real-time approaches.

Two different ADL datasets have been used to evaluate our proposal. Ordóñez dataset [53] was developed in the UC Irvine Machine Learning Repository. It represents two participants performing ten ADL activities in their own homes. The activities were performed individually, and this dataset is composed by two instances of data: Room A and Room B, each one corresponding to a different user and summing up to 35 days. Ten activities were classified: *breakfast*, *dinner*, *leaving*, *lunch*, *showering*, *sleeping*, *snack*, *spare time TV*, and *grooming*. The number of sensors was 12, although two of them were never fired in the case of the second participant. In fact, the dataset can be actually considered as two different datasets. The Room B dataset was chosen because an exploratory analysis shows that the activities in this set are more difficult to be recognized, so there are more chances to find statistically significant differences. However, this dataset is particularly imbalanced, with some of the activities having less than fifteen annotated examples, which hinders the adoption of a complex data partitioning scheme for validating the experiment. Only one set of examples has been used to train and validate the proposal, so some overfitting is expected.

The other dataset, Singla et al. [57], represents a sensor data stream registered in the Washington State University smart apartment. The data represents twenty participants performing eight ADL activities. These activities were conducted 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. 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 framework proposed in Section 4 has been used to convert both datasets to all the available ontology models for the description of ADL, namely, Salguero [3], SPHERE [21], COSAR [22] and Noor [23]. In addition, a version with the *OMA* ontology has been generated in which information about the context has been included, following the scheme proposed in Section 4.3. All the instances of the activities in the datasets have been created as individuals of the generic activity concept of each ontology. Therefore, there is no atomic concept A in the ontologies that can be returned by the CEL algorithms as a solution.

Before discussing the results obtained in the evaluation, it is important to note that all the ontologies proposed by other authors have been slightly modified in order to be able to apply the CEL technique. Some of the ontologies have required very subtle changes. Others have had to be modified in greater depth. These changes are necessary because the ontologies have not been designed to make the reasoning mechanisms efficient but to be as much expressive as possible. This means that the data mining techniques, such as the one employed in this experiment, cannot be directly applied to them because they are usually based on the intensive use of reasoning mechanisms. None of the original ontologies can obtain results after one hour of searching.

The ontology proposed by Noor is the most similar ontology to the one proposed in this paper. In this model, the activities are composed of smaller time intervals. These time intervals are those which are really associated with the events generated by the sensors, in such a way that we can characterize an interval according to the sensors that are activated during it. The activities are then defined as sequences of intervals in which certain sensors are fired. More than eighty thousand intervals of thirty-second lengths are required to represent all the activities in the datasets, which is impossible for a reasoner to handle. Instead, we divided each instance of each activity window into three different intervals, which produces a reasonable number of individuals in the ontology. Despite this change, the reasoners waste several hours to process the optimized ontologies generated from the datasets in the experiment. The solution to this problem has been to define as disjoint all the concepts that represent the different sensors and types of activities in the datasets. This action is done automatically in the plugin specifically developed for the "Ontology converter" application that generates the ontology in the Noor format.

The SPHERE ontology does not present performance problems. The problem with this model is that it does not produce relevant results. The reason is that there is no property that allows reasoners to establish a relationship between the activities and the events that occur within them. This problem has been resolved by setting the concept *Activity* as the domain of the property *has*. In addition, to increase the probability of this relationship being used, the inverse property of the *has* property has been made explicit by creating a new named property.

The most important problem of the COSAR ontology is that it contains multiple properties to relate activities to other entities of the ontology. Many of them only have sense after the rules for determining the activity that is being performed have been found (such as the property *necessarySensorEventFor*). Furthermore, most of them do not keep a strong relation with ADL, such as the property *hasStudent*. However, all these properties must be considered by the CEL algorithm, making the search space grow exponentially and degrading the performance of the algorithm.

Moreover, according to the text of the work in which the COSAR ontology is proposed, the property *occursIn* should relate the events to the activities in which they occur. In the public version of the ontology, this property relates the activities to the apartments in which they are carried out, as well as the property *activityOccursInApt* does. We understand that it is an erratum and therefore the range of the property *occursIn* has been modified to refer to the *Activity* concept. As in the SPHERE model, the inverse properties of the properties *occursIn* and *producesEvent*, which relate the

events to the activities and the events with the sensors, respectively, have also been made explicit. All the individuals in the ontology have been removed, and the *SensorEvent* and *Artifact* concepts have been defined as disjoint concepts for the sake of efficiency. The *Artifact* concept encompasses all objects in the smart environment, including sensors.

To determine the activity being performed at each instant, the CEL technique, described in Section 3, has been applied. The DL-Learner application [58], an open source application that implements some CEL algorithms, has been used for this task. More precisely, we have employed the CELOE and the OCEL algorithms implementation in the experiment [59], with 0% of noise percentage and FastInstanceChecker (FIC) as the reasoner. The FIC reasoner is a special kind of reasoner, specifically developed for the DL-Learner application that basically makes some violations of the open world assumptions in OWL to improve the efficiency of the reasoning mechanism. This is very convenient in the context of CEL algorithms because of the extremely high number of class descriptions that have to be evaluated in the process.

6. Results and Discussion

Table 3 shows the accuracy ($C^+/(C^+ + C^-)$) of the class expressions obtained by the DL-Learner application to classify all the activities of the two datasets used in the experiment. The implementation of the OCEL algorithm only generates information about the accuracy. The implementation of the CELOE algorithm also generates the *f*-measure $((2 \cdot \text{accuracy} \cdot \text{recall})/(\text{accuracy} + \text{recall}))$, which is a more representative measure of the quality of the classifiers. The values of the *f*-measure obtained with the CELOE algorithm are shown in Table 4. In all cases, the DL-Learner application has been configured to stop the search for new class expressions after five minutes.

The best overall result is obtained using the Salguero ontology and the OCEL algorithm, with an average accuracy of 92.04%. There is no significant statistical difference with respect to the result obtained with the OMA ontology which obtains a slightly lower average accuracy (90.07%) with the same algorithm. The difference between these two cases is more noticeable in the activities corresponding to the Ordoñez dataset. Activities such as *Breakfast* or *Sleeping* are much better recognized when the OMA ontology is used, while activities such as *Toileting*, *Dinner*, and *Spare time TV* are better recognized using the Salguero ontology. As an example, the class descriptions (3) and (4) in Table 5 show the best descriptions found with the OCEL algorithm for the *Breakfast* activity. The class description (3), obtained using the OMA ontology, describes the activity *Breakfast* as the activity that begins and ends with the activation of the kitchen door sensor or in which the sensor of the fridge is activated and the third of the recorded events is produced by a sensor placed on a door. In addition, the *Breakfast* activity never ends with the activation of the main door sensor and must always include the activation, at some instant, of the kitchen door sensor. The class description (4), obtained with the Salguero ontology, describes the activity *Breakfast* as the activity in which the door sensor in the living room is never fired after the kitchen door sensor is activated. The activity cannot start with the activation of the main door sensor. As can be seen, the sensor of the main door is, somewhat surprisingly, relevant to determine if the *Breakfast* activity is happening or not in both cases. It is also important to point out that, in the case of the Salguero ontology, the order in which the events occur is more relevant, imposing in this case that the door sensor in the living room must not be activated after the kitchen door sensor. In the case of the OMA ontology, however, contextual information has been introduced in the description of the *Breakfast* activity, considering the type of sensor that should produce the event but not the specific sensor. This information was not available in the Salguero ontology and is one of the contributions of this work.

Actually, the Noor ontology is the ontology with which better description is found for the *Breakfast* activity, with an accuracy of 97.43% for both CEL algorithms. However, the class description found by both algorithms for the description of the *Breakfast* activity (description (6) in Table 5) does not seem to provide much relevant information, apart from restricting the duration of the activity between five and ten minutes. Nevertheless, given the high value of the *f*-measure, we may consider the interval as a key feature of the activity. The same happens with the *Grooming* and *Spare time TV* activities, for which high values of the *f*-measure are also obtained with the Noor ontology. In view of the results, the typical duration of the *Grooming* activity is between one and five minutes (description (7) in Table 5), while the *Spare time TV* activity usually exceeds ten minutes (description (8) in Table 5).

As can be seen in Figure 5, the OCEL algorithm always reports better accuracy than the CELOE algorithm. However, it is important to remember that the implementation of the OCEL algorithm does not provide information about the *f*-measure. The accuracy value represents how many of the activities described by the generated class expression actually correspond to the activity being described, but it does not provide information about how many of the activities that should also be recognized are described by the generated class expression. The *f*-measure is a more relevant measure in the field of supervised learning, since it includes both accuracy and completeness. The CELOE algorithm is also considered in the experiment for this reason.

The accuracy values obtained with the CELOE algorithm are very similar to those obtained with the OCEL algorithm. The best results are achieved when the OMA and the Salguero ontologies are employed (Figure 5). The same situation occurs when the results are analyzed from the *f*-measure point of view (Figure 6). The results obtained with the OMA and the Salguero ontologies are generally better than those obtained with the rest of the ontologies. Although the difference is not statistically significant ($p < 0.1$), we can see that the results obtained with the Salguero ontology are slightly better than those obtained with the OMA ontology. In fact, there is not a single activity in which the results obtained with the OMA ontology are better than those obtained with the Salguero ontology. Considering that the

Table 3: Accuracies of class descriptions found by the CEL algorithms for both datasets.

Dataset	Activity	CELOE					OCEL				
		COSAR	Noor	OMA	Salguero	SPHERE	COSAR	Noor	OMA	Salguero	SPHERE
Singla et al. [57]	Answer the phone	12.57	85.11	51.50	51.50	33.53	92.22	85.11	98.80	98.80	89.22
	Choose outfit	99.40	85.11	100.00	100.00	100.00	100.00	85.28	100.00	100.00	100.00
	Clean	79.64	73.76	95.21	97.60	95.21	99.40	82.62	100.00	99.40	99.40
	Fill medication dispenser	46.11	51.60	97.01	97.01	94.01	46.11	65.43	96.41	100.00	77.25
	Prepare birthday card	90.42	60.99	92.81	94.61	90.42	94.01	71.10	98.20	98.20	97.60
	Prepare soup	96.41	55.85	99.40	100.00	96.41	100.00	76.95	100.00	100.00	100.00
	Wash DVD	95.81	52.13	97.60	98.80	95.81	99.40	64.01	98.80	100.00	99.40
	Water plants	83.23	100.00	90.42	90.42	88.02	88.02	100.00	91.02	95.21	97.60
Ordoñez et al. [53]	Breakfast	57.56	97.43	61.56	74.00	57.56	81.56	97.43	87.78	77.11	57.66
	Dinner	40.00	94.86	72.00	72.00	67.33	51.56	94.86	81.11	90.44	56.67
	Grooming	19.56	90.30	41.11	82.00	41.11	49.33	90.30	65.11	64.44	49.33
	Leaving	93.78	67.17	94.67	96.67	93.78	94.00	67.17	94.89	99.56	94.00
	Lunch	55.56	60.57	75.11	81.56	55.56	72.67	70.50	87.78	94.22	72.67
	Showering	96.89	71.38	100.00	100.00	96.89	100.00	78.86	100.00	100.00	100.00
	Sleeping	68.89	63.38	79.33	80.89	68.89	72.22	72.37	94.89	90.67	68.89
	Snack	62.89	77.80	66.44	79.33	62.89	80.44	83.47	81.11	82.44	62.89
	Spare time TV	61.78	83.76	62.44	66.44	61.78	61.78	83.76	72.44	80.00	61.78
	Toileting	19.33	100.00	57.33	57.33	55.11	71.33	100.00	72.89	86.22	69.11
	Average	65.55	76.18	79.66	84.45	75.24	80.78	81.62	90.07	92.04	80.75

Table 4: *f*-Measure of class descriptions found by the CELOE algorithm for both datasets.

Dataset	Activity	CELOE				
		COSAR	Noor	OMA	Salguero	SPHERE
Singla et al. [57]	Answer the phone	22.34	66.67	34.15	34.15	27.45
	Choose outfit	97.67	66.67	100.00	100.00	100.00
	Clean	55.26	45.99	84.00	91.30	84.00
	Fill medication dispenser	30.77	30.53	88.89	88.89	80.00
	Prepare birthday card	72.41	36.42	77.78	82.35	72.41
	Prepare soup	87.50	40.29	97.67	100.00	87.50
	Wash DVD	85.71	31.82	91.30	95.45	85.71
	Water plants	60.00	100.00	72.41	72.41	67.74
Ordoñez et al. [53]	Breakfast	18.72	80.00	20.28	27.33	18.72
	Dinner	7.53	50.00	14.86	14.86	13.02
	Grooming	32.71	80.92	39.91	68.48	39.91
	Leaving	73.08	35.10	76.00	83.52	73.08
	Lunch	11.50	10.36	18.84	23.85	11.50
	Showering	50.00	10.26	100.00	100.00	50.00
	Sleeping	29.29	21.72	38.41	40.28	29.29
	Snack	35.52	42.07	37.86	49.73	35.52
	Spare time TV	55.90	75.83	56.33	59.08	55.90
	Toileting	32.40	100.00	47.54	47.54	46.28
	Average	47.68	51.37	60.90	65.51	54.34

main difference between both ontologies is that in the OMA ontology also includes information about the context of the activities, we could deduce that the sequence in which the events occur is more relevant to determine the type of activity that is happening, instead of other information such as the type of sensor or its location. Although the OMA ontology also includes information about the order in which events occur, the search space for the CEL algorithms is much larger in the case of the OMA ontology. On the contrary, the CEL algorithm spends the entire execution time just testing different sequences of events when the Salguero ontology is used.

The importance of the order of events is clearly reflected in the case of the *Showering* activity. The CELOE algorithm manages to find a class expression that perfectly defines this activity when the OMA and Salguero ontologies are used. In both cases, the *Showering* activity is defined as the activity that ends with the activation of the shower sensor (description (9) in Table 5). The fact that this sensor is the last to be activated during the activity is very relevant, since when this restriction in the order is not imposed, the *f*-measure value decreases to 50%, as in the case of the COSAR ontology and SPHERE (see descriptions (10) and (11)). The duration of this activity is also irrelevant. The class expression

Table 5: Best descriptions found for some activities.

Class description		Explanation
(((endsWith some DoorKitchen) and (startsWith some DoorKitchen)) or ((has some FridgeKitchen) and (startsWith some (hasNext some (hasNext some (producedBy some DoorSensor)))))) and (endsWith some (not (MaindoorEntrance))) and (has some DoorKitchen)	(3)	*Breakfast (OMA-OCEL)*: an activity that starts and ends with the activation of the kitchen door sensor or an activity in which the fridge has been opened and the third event has been fired by a door sensor. In any case, the sensor in the main entrance door cannot be the last sensor activated and the sensor in the kitchen door has to be activated at some instant during the activity.
(has some (DoorKitchen and (isFollowedBy max 1 (not (DoorLiving))))) and (startsWith some (not (MaindoorEntrance)))	(4)	*Breakfast (Salguero-OCEL)*: an activity in which the living door sensor has not been activated after the kitchen door has been opened. The activity cannot start with the activation of the main entrance door.
((startsWith some (isFollowedBy only ShowerBathroom)) and (startsWith only DoorKitchen)) or (hasItem some BasinBathroom)	(5)	*Grooming (Salguero-CELOE)*: an activity in which the living door sensor has not been activated after the kitchen door has been opened. The activity cannot start with the activation of the main entrance door.
LongerThan5minutes and (not (LongerThan10minutes))	(6)	*Breakfast (Noor-CELOE)*: an activity that lasts between 5 and 10 minutes.
LongerThan1minute and (not (LongerThan5minutes))	(7)	*Grooming (Noor-CELOE)*: an activity that lasts between 1 and 5 minutes.
LongerThan10minutes	(8)	*Spare time TV (Noor-CELOE)*: an activity that lasts more than 10 minutes.
endsWith only ShowerBathroom	(9)	*Showering (OMA-CELOE)*: an activity that always ends with an activation of the shower sensor.
contains some (isProducedBy only ShowerBathroom)	(10)	*Showering (COSAR-CELOE)*: an activity that contains an event produced by the shower sensor.
inverseHas some ShowerBathroom	(11)	*Showering (SPHERE-CELOE)*: an activity that contains an event produced by the shower sensor.
LongerThan1minute and (not (LongerThan5minutes))	(12)	*Showering (Noor-CELOE)*: an activity that lasts between 1 and 5 minutes.

produced by the algorithm when the Noor ontology is used, shown in the description (12) in Table 5, barely achieves an *f*-measure value of above 10%.

It is also important to note that the order of the events is not always relevant for the recognition of some of the activities. In the case of the *Grooming* activity, for example, its duration is much more relevant than the activation of a specific sensor. In fact, the class description (7), also obtained after applying the CELOE algorithm to the Noor ontology, obtains a much higher *f*-score value (80.92%) than the one obtained when the Salguero ontology is used (see description (5) in Table 5), which considers the activation of the basin sensor, mainly.

In view of the results, we can conclude that the Salguero ontology is in general the most appropriate ontology for the application of CEL techniques for the recognition of ADL. The OMA ontology, whose core is based on the Salguero ontology, also obtains very good results in general but slightly lower. The CEL algorithms obtain worse results in general when using the other ontologies although it is also important to highlight the fact that the Noor ontology produces the best descriptions for some of the activities, such as the *Water plants* and *Toileting* activities, for which perfect descriptions are generated. In all the activities in which the best results are achieved using the Noor ontology, the CEL algorithms end up finding that the best way to describe these activities is according to their duration. The *Water plants* and *Toileting* activities are described as activities whose duration is less than thirty seconds and one minute, respectively (they are in different datasets). The activity *Grooming* usually lasts between one and five minutes, while the *Breakfast* and *Dinner* activities usually last between five

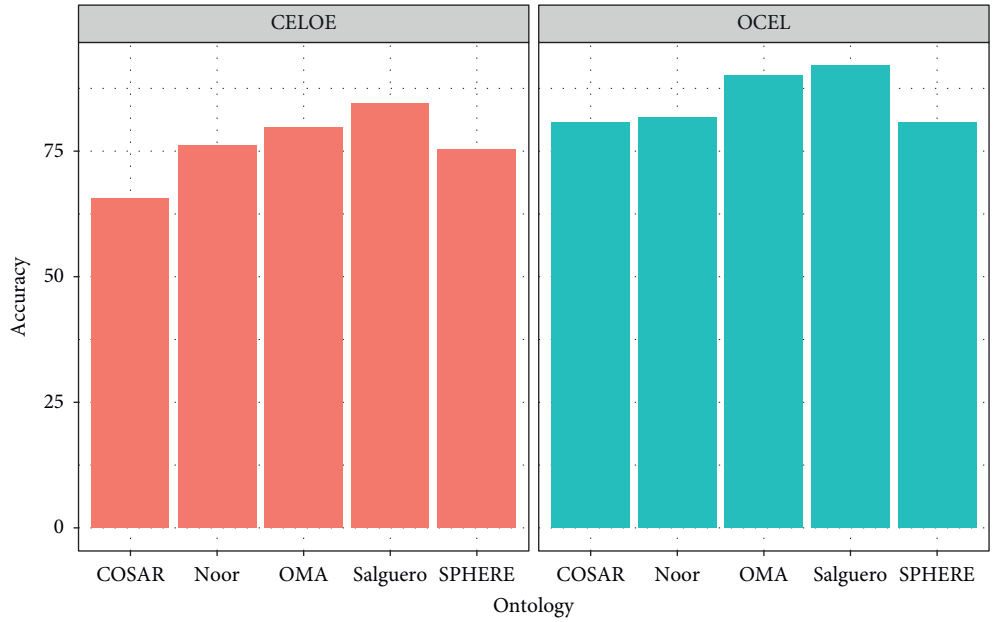

Figure 5: Accuracy vs ontology.

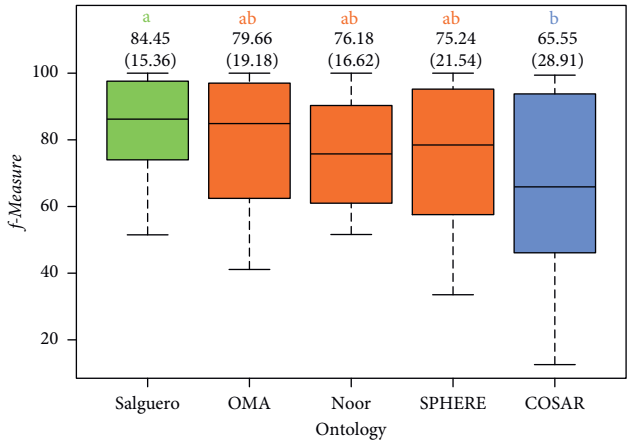

Figure 6: Ontology vs *f*-measure for CELOE algorithm.

and ten minutes. The *Spare time TV* activity is the only one with a duration longer than ten minutes. The Noor ontology produces better results in these cases because it is the only one that makes this information explicit.

Since the SPHERE ontology follows an approach similar to the OMA and Salguero ontologies, its behavior is analogous to these proposals. Although the results obtained with the SPHERE ontology do not improve any of the results obtained with the OMA and Salguero ontologies, the activity descriptions found with the SPHERE ontology are good, in general.

7. Conclusion

The proliferation of smart homes has led to multiple proposals for the automatic recognition of ADL, being many of them based on the use of ontologies. In this work, we have evaluated the CEL technique as a mechanism for the automatic recognition of activities. For this, we have developed a framework that allows the loading of many well-known datasets, as well as the conversion of these datasets to five different ontologies available in the literature for the representation of ADL. The development of this framework has included the extension of an ontology that we previously proposed for the recognition of ADL. This ontology, unlike the others in the literature, was designed to solely describe activities as sequences of sensor events, without considering more information about the context in which they happen. The rest of the ontologies available in the literature use the opposite approach, describing the activities according to the sensors that are activated, their type, and location, as well as information about the people who perform them. In this work, we have extended the previously proposed ontology to also consider the contextual information available in the rest of the ontologies.

Once the framework was developed, an experiment was carried out to determine the suitability of the different ontologies for the representation of ADL and the performance of two CEL algorithms for their automatic recognition. In view of the results, we can conclude that the order in which events occur is more relevant for most of the activities than the rest of the contextual information. CEL algorithms end up generating much more representative class expressions when ontologies that consider the sequences of events are used. In fact, the best overall results are obtained with the ontology that only describes the ADL as sequences of events and does not include any other additional information. Furthermore, when the algorithms employ the extended version of that ontology, which also includes contextual information, the results are still class expressions in which the order of events is more relevant. There are, however, some activities for which this rule is not met. In those cases, the best results have been obtained with the Noor ontology, which is the only one

that represents the duration of the activities explicitly. In fact, the duration of the activities is a key feature to recognize activities such as *Breakfast*, *Toileting*, and *Answer the phone*.

In view of this result, we are currently working on the incorporation of this information in the extended version of the ontology. Our goal is to find the most efficient way to make this information explicit using automatic reasoning, without having to annotate it when generating the ontology from the dataset.

Data Availability

The data used to support the findings of this study were made available by their corresponding authors in the UC Irvine Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Activities+of+Daily+Living+%28ADLs%29+Recognition+Using+Binary+Sensors) and the Center for Advances Studies in Adaptive Systems Repository (http://casas.wsu.edu/datasets/adlinterweave.zip), as described in Section 5. For the sake of completeness, we decided to also make available all the ontologies that were generated in the experiment from the original data along with the source code of the applications in the framework proposed in this work. They are all available at https://sourceforge.net/p/adl-miningframework/code/HEAD/tree/trunk/tests/.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the REMIND project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under the Marie Skłodowska-Curie grant agreement no. 734355.

References

[1] L. Chen, J. Hoey, C. Nugent, D. Cook, and Z. Yu, "Sensor-based activity recognition," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 6, pp. 790–808, 2012.

[2] F. Quesada, F. Moya, J. Medina, L. MartÃnez, C. Nugent, and M. Espinilla, "Generation of a partitioned dataset with single, interleave and multioccupancy daily living activities," in *Proceedings of International Conference on Ubiquitous Computing and Ambient Intelligence*, vol. 9454, pp. 60–71, Puerto Varas, Chile, December 2015.

[3] A. Salguero, J. Medina, P. Delatorre, and M. Espinilla, "Methodology for improving classification accuracy using ontologies: application in the recognition of activities of daily living," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–18, 2018, In press.

[4] C. Li, M. Lin, L. T. Yang, and C. Ding, "Integrating the enriched feature with machine learning algorithms for human movement and fall detection," *Journal of Supercomputing*, vol. 67, no. 3, pp. 854–865, 2014.

[5] D. Riboni, T. Sztyler, G. Civitarese, and H. Stuckenschmidt, "Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning," in *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 1–12, ACM, Heidelberg, Germany, September 2016.

[6] L. Chen, C. D. Nugent, and H. Wang, "A knowledge-driven approach to activity recognition in smart homes," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 6, pp. 961–974, 2012.

[7] I.-H. Bae, "An ontology-based approach to ADL recognition in smart homes," *Future Generation Computer Systems*, vol. 33, pp. 32–41, 2014.

[8] A. Salguero, M. Espinilla, P. Delatorre, and J. Medina, "Using ontologies for the online recognition of activities of daily living," *Sensors*, vol. 18, no. 4, p. 1202, 2018.

[9] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What are ontologies, and why do we need them?," *IEEE Intelligent Systems*, vol. 14, no. 1, pp. 20–26, 1999.

[10] M. Uschold and M. Gruninger, "Ontologies: principles, methods and applications," *The Knowledge Engineering Review*, vol. 11, no. 2, pp. 93–136, 1996.

[11] J. Knijff, F. Frasincar, and F. Hogenboom, "Domain taxonomy learning from text: the subsumption method versus hierarchical clustering," *Data & Knowledge Engineering*, vol. 83, pp. 54–69, 2013.

[12] F. Baader, D. Calvanese, D. McGuinness, P. Patel-Schneider, and D. Nardi, *The Description Logic Handbook: Theory, Implementation and Applications*, Cambridge University Press, Cambridge, UK, 2003.

[13] T. Wei, Y. Lu, H. Chang, Q. Zhou, and X. Bao, "A semantic approach for text clustering using WordNet and lexical chains," *Expert Systems with Applications*, vol. 42, no. 4, pp. 2264–2275, 2015.

[14] I. Horrocks, "Ontologies and the semantic web," *Communications of the ACM*, vol. 51, no. 12, pp. 58–67, 2008.

[15] J. Köhler, S. Philippi, M. Specht, and A. Rüegg, "Ontology based text indexing and querying for the semantic web," *Knowledge-Based Systems*, vol. 19, no. 8, pp. 744–754, 2006.

[16] A. Maedche and S. Staab, "Ontology learning for the semantic web," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 72–79, 2001.

[17] S. De, P. Barnaghi, M. Bauer, and S. Meissner, "Service modelling for the internet of things," in *Proceedings of 2011 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 949–955, IEEE, Szczecin, Poland, September 2011.

[18] S. Hachem, T. Teixeira, and V. Issarny, "Ontologies for the internet of things," in *Proceedings of the 8th Middleware Doctoral Symposium*, p. 3, ACM, Lisbon, Portugal, December 2011.

[19] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen, "From SHIQ and RDF to OWL: the making of a web ontology language," *Journal of Web Semantics*, vol. 1, no. 1, pp. 7–26, 2003.

[20] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: a practical owl-dl reasoner," *Journal of Web Semantics*, vol. 5, no. 2, pp. 51–53, 2007.

[21] P. Woznowski, R. King, W. Harwin, and I. Craddock, "Activity recognition framework for healthcare applications: ontology, labelling strategies, and best practice," in *Proceedings of International Conference on Internet of Things and Big Data*, pp. 369–377, Rome, Italy, April 2016.

[22] D. Riboni and C. Bettini, "COSAR: hybrid reasoning for context-aware activity recognition," *Personal and Ubiquitous Computing*, vol. 15, no. 3, pp. 271–289, 2011.
[23] M. H. M. Noor, Z. Salcic, I. Kevin, and K. Wang, "Ontology-based sensor fusion activity recognition," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–15, 2018, In press.
[24] L. Chen and C. Nugent, "Ontology-based activity recognition in intelligent pervasive environments," *International Journal of Web Information Systems*, vol. 5, no. 4, pp. 410–430, 2009.
[25] J. Hois, "Modularizing spatial ontologies for assisted living systems," in *Proceedings of International Conference on Knowledge Science, Engineering and Management*, pp. 424–435, Springer, Belfast, UK, September 2010.
[26] I. McChesney, C. Nugent, J. Rafferty, and J. Synnott, "Exploring an open data initiative ontology for shareable smart environment experimental datasets," in *Proceedings of International Conference on Ubiquitous Computing and Ambient Intelligence*, pp. 400–412, Springer, Philadelphia, PA, USA, November 2017.
[27] R. Tan, J. Gu, Z. Zhong, and P. Chen, "Socom: multi-sensor oriented context model based on ontologies," in *Proceedings of 2012 8th International Conference on Intelligent Environments (IE)*, pp. 236–242, IEEE, Guanajuato, Mexico, June 2012.
[28] G. Baryannis, P. Woznowski, and G. Antoniou, "Rule-based real-time adl recognition in a smart home environment," in *Proceedings of International Symposium on Rules and Rule Markup Languages for the Semantic Web*, pp. 325–340, Springer, Stony Brook, NY, USA, July 2016.
[29] M. Botts and A. Robin, "OpenGIS sensor model language (sensorml) implementation specification," in *OpenGIS Implementation Specification*, Vol. 7, OGC, Boston, MA, USA, 2007.
[30] K. S. Gayathri, K. S. Easwarakumar, and S. Elias, "Probabilistic ontology based activity recognition in smart homes using Markov logic network," *Knowledge-Based Systems*, vol. 121, pp. 173–184, 2017.
[31] L. Chen, C. Nugent, and G. Okeyo, "An ontology-based hybrid approach to activity modeling for smart homes," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 1, pp. 92–105, 2014.
[32] I. Horrocks, B. Motik, and Z. Wang, "The hermit owl reasoner," in *Proceedings of CEUR Workshop*, Lyon, France, April 2012.
[33] S.-H. Nienhuys-Cheng and R. De Wolf, *Foundations of Inductive Logic Programming*, Springer Science & Business Media, vol. 1228, Berlin, Germany, 1997.
[34] A. Maedche and S. Staab, "Ontology learning for the semantic web," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 72–79, 2001.
[35] J. Lehmann and J. Voelker, "An introduction to ontology learning," in *Perspectives on Ontology Learning*, IOS Press, Amsterdam, The Netherlands, 2014.
[36] C. Klaussner and D. Zhekova, "Lexico-syntactic patterns for automatic ontology building," in *Proceedings of the Second Student Research Workshop associated with RANLP 2011*, pp. 109–114, Hissar, Bulgaria, September 2011.
[37] N. Sanchez-Pi, L. Martí, and A. C. Bicharra Garcia, "Improving ontology-based text classification: an occupational health and security application," *Journal of Applied Logic*, vol. 17, pp. 48–58, 2016.
[38] P. Ristoski, C. Bizer, and H. Paulheim, "Mining the web of linked data with rapidminer," *Journal of Web Semantics*, vol. 35, pp. 142–151, 2015.
[39] L. Bühmann, J. Lehmann, and P. Westphal, "DL-learner—a framework for inductive learning on the Semantic Web," *Journal of Web Semantics*, vol. 39, pp. 15–24, 2016.
[40] R. Meng, Y. Tong, L. Chen, and C. C. Cao, "Crowdtc: crowdsourced taxonomy construction," in *Proceedings of 2015 IEEE International Conference on Data Mining (ICDM)*, pp. 913–918, IEEE, Atlantic City, NJ, USA, November 2015.
[41] J. Lehmann and P. Hitzler, "Concept learning in description logics using refinement operators," *Machine Learning*, vol. 78, no. 1-2, pp. 203–250, 2010.
[42] J. Lehmann, S. Auer, L. Bühmann, and S. Tramp, "Class expression learning for ontology engineering," *Journal of Web Semantics*, vol. 9, no. 1, pp. 71–81, 2011.
[43] A. Tran, J. Dietrich, H. Guesgen, and S. Marsland, "An approachto parallel class expression learning," in Rules on the Web: Research and Applications, vol. 7438 of Lecture Notes in Computer Science, A. Bikakis and A. Giurca, Eds., pp. 302–316, Springer Berlin Heidelberg, Berlin, Germany, 2012.
[44] L. Badea and S.-H. Nienhuys-Cheng, "A refinement operator for description logics," in *Proceedings of International Conference on Inductive Logic Programming*, pp. 40–59, Springer, London, UK, July 2000.
[45] L. Iannone, I. Palmisano, and N. Fanizzi, "An algorithm based on counterfactuals for concept learning in the semantic web," *Applied Intelligence*, vol. 26, no. 2, pp. 139–159, 2007.
[46] N. Fanizzi, C. d'Amato, and F. Esposito, "DL-foil in description logics," in *Proceedings of International Conference on Inductive Logic Programming*, pp. 107–121, Springer, Prague, Czech Republic, September 2008.
[47] U. Straccia and M. Mucci, "pFOIL-DL: learning (fuzzy) el concept descriptions from crisp owl data using a probabilistic ensemble estimation," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, pp. 345–352, ACM, Salamanca, Spain, April 2015.
[48] G. Rizzo, N. Fanizzi, C. d'Amato, and F. Esposito, "A framework for tackling myopia in concept learning on the web of data," in *Proceedings of European Knowledge Acquisition Workshop*, pp. 338–354, Springer, Nancy, France, November 2018.
[49] A. R. Divroodi and L. A. Nguyen, "On bisimulations for description logics," *Information Sciences*, vol. 295, pp. 465–493, 2015.
[50] T.-L. Tran, Q.-T. Ha, T.-L.-G. Hoang, L. A. Nguyen, and H. S. Nguyen, "Bisimulation-based concept learning in description logics," *Fundamenta Informaticae*, vol. 133, no. 2-3, pp. 287–303, 2014.
[51] J. Lehmann, *Learning OWL Class Expressions*, vol. 22, IOS Press, Amsterdam, Netherlands, 2010.
[52] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of the 10th International Conference on Ubiquitous Computing (UbiComp 2008)*, pp. 1–9, Seoul, Korea, September 2008.
[53] F. J. Ordóñez, P. de Toledo, and A. Sanchis, "Activity recognition using hybrid generative/discriminative models on home environments using binary sensors," *Sensors*, vol. 13, no. 5, pp. 5460–5477, 2013.
[54] M. Espinilla, J. Medina, J. Hallberg, and C. Nugent, "A new approach based on temporal sub-windows for online sensor-based activity recognition," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–13, 2018, In press.
[55] A. Sixsmith and N. Johnson, "A smart sensor to detect the falls of the elderly," *IEEE Pervasive Computing*, vol. 3, no. 2, pp. 42–47, 2004.

[56] I.-K. Hwang and J.-W. Baek, "Wireless access monitoring and control system based on digital door lock," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 4, pp. 1724–1730, 2007.
[57] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Tracking activities in complex settings using smart environment technologies," *International Journal of Biosciences, Psychiatry, and Technology*, vol. 1, no. 1, p. 25, 2009.
[58] J. Lehmann, "DL-learner: learning concepts in description logics," *Journal of Machine Learning Research*, vol. 10, pp. 2639–2642, 2009.
[59] A. C. Tran, J. Dietrich, H. W. Guesgen, and S. Marsland, "Parallel symmetric class expression learning," *Journal of Machine Learning Research*, vol. 18, no. 64, pp. 1–34, 2017.