# POLARIS: Probabilistic and Ontological Activity Recognition in Smart-homes

Gabriele Civitarese, Timo Sztyler, Daniele Riboni, Claudio Bettini, Heiner Stuckenschmidt

**Abstract**—Recognition of activities of daily living (ADLs) is an enabling technology for several ubiquitous computing applications. Most activity recognition systems rely on supervised learning to extract activity models from labeled datasets. A problem with that approach is the acquisition of comprehensive activity datasets, which is an expensive task. The problem is particularly challenging when focusing on complex ADLs characterized by large variability of execution. Moreover, several activity recognition systems are limited to offline recognition, while many applications claim for online activity recognition. In this paper, we propose POLARIS, a framework for unsupervised activity recognition. POLARIS can recognize complex ADLs exploiting the semantics of activities, context data, and sensors. Through ontological reasoning, our algorithm derives semantic correlations among activities and sensor events. By matching observed events with semantic correlations, a statistical reasoner formulates initial hypotheses about the occurred activities. Those hypotheses are refined through probabilistic reasoning, exploiting semantic constraints derived from the ontology. Our system supports online recognition, thanks to a novel segmentation algorithm. Extensive experiments with real-world datasets show that the accuracy of our unsupervised method is comparable to the one of supervised approaches. Moreover, the online version of our system achieves essentially the same accuracy of the offline version.

**Index Terms**—Ontological reasoning, Probabilistic reasoning, Online/Offline Activity recognition, Unsupervised classification, Pervasive computing

## 1 Introduction

The rapid growing of the population age in industrialized societies calls for advanced tools to continuously monitor the activities of elderly people at home. The goals of those tools are to support active and healthy ageing, and to early detect possible health issues. Recent advancements in sensor miniaturization and wireless communications have paved the way to unobtrusive activity recognition systems. Those systems can continuously monitor activities of daily living (ADLs) based on low-level behaviors (e.g., moving to the kitchen or opening a drawer) and artificial intelligence methods. Unfortunately, while those systems are effective in controlled environments, their actual effectiveness out of the lab is still limited [1] due to different shortcomings of existing approaches.

Currently, most activity recognition systems rely on supervised learning applied to datasets of activities and sensor data [2], [3]. Supervised learning proves to be effective in recognizing activities characterized by specific postures or motions, such as physical activities. However, its applicability to complex ADLs (e.g., cooking, cleaning, and dressing) is problematic. On the one side, the way in which individuals perform ADLs strongly depends on current context conditions. Hence, large datasets of ADLs must be acquired to capture most execution patterns in different situations. On the other side, activity execution patterns are strongly coupled to the individual's characteristics and home environment, and the portability of activity datasets is an open issue [4]. Therefore, ideally one extensive ADLs dataset should be acquired from each monitored individual. Unfortunately, acquiring ADLs datasets is expensive in terms of annotation costs [5], [6]. Besides, activity annotation by an external observer, by means of cameras or direct observation, violates the user's privacy.

To avoid the burden of dataset acquisition, other works rely on knowledge-based activity models, manually specified through logic languages and ontologies. Those models are matched with acquired sensor data to recognize the activities [7]. A major shortcoming of that approach is the rigidity of specifications. For instance, complex ADLs are often specified through temporal sequences of simpler actions [8]. Nevertheless, it is unfeasible to enumerate all the possible sequences of actions describing a complex ADL.

Moreover, several ambient intelligence applications call for *online* activity recognition systems; i.e., systems that can recognize the current activity in nearly real-time [9]. For instance, a system to detect dangerous behaviors of the elderly should report the potential danger as it happens, since a delay could put the elderly's safety at risk. Unfortunately, several ADL recognition systems are limited to offline recognition, and the accuracy of real-time ADL recognition systems is generally lower than those of offline ones [10].

In this work, we propose a novel framework for Probabilistic and OntoLogical Activity RecognItion in Smart-homes (POLARIS) to overcome the main limitations of existing ADL recognition systems. Our method is unsupervised: by reasoning with an OWL 2 ontology [11] that models activities and smart home infrastructure, we mine probabilistic semantic correlations among sensor events and activities. We translate our ontological model into a Markov Logic Network (MLN) [12], and we perform probabilistic and knowledge-based reasoning for segmenting and recognizing the activities. With respect to other reasoning frameworks, our MLN-based solution has the following advantages: (i) it supports probabilistic reasoning through soft rules,

*G. Civitarese and C. Bettini are with the EveryWare Lab, University of Milan, Via Celoria 18, I-20133 Milano, Italy*
*e-Mail: (gabriele.civitarese | claudio.bettini)@unimi.it*

*T. Sztyler and H. Stuckenschmidt are with the University of Mannheim, B6 26, D-68159 Mannheim, Germany*
*e-Mail: (timo | heiner)@informatik.uni-mannheim.de*

*D. Riboni is with the University of Cagliari, Via Ospedale 72, I-09124 Cagliari, Italy*
*e-Mail: riboni@unica.it*

(ii) it supports the definition of deterministic ontological axioms through hard rules, (iii) thanks to an extension of MLN with numerical constraints, the MLN reasoning framework that we adopt ($MLN_{NC}$) supports temporal reasoning, and (iv) $MLN_{NC}$ is supported by an optimized reasoner. The use of a probabilistic logic is also motivated by the need to deal with noisy sensor data. Moreover, our MLN model is carefully crafted to support recognition of interleaved activities.

An important feature of POLARIS is the ability to support both online and offline ADL recognition, in order to cope with different application requirements. Indeed, thanks to the online sensor data segmentation algorithm of POLARIS, applications having real-time requirements can trade a small amount of accuracy for real-time recognition capabilities. On the contrary, other applications (e.g., systems for long-term behavior monitoring) can exploit the offline version of our framework.

We performed extensive experiments with real-world datasets of ADLs performed by twenty-two individuals in two different smart-home environments. Overall, the results showed that, even using a smaller number of sensors, the performance of our unsupervised method is comparable to the one of state-of-the-art supervised algorithms. Compared to other unsupervised approaches, experiments have shown that our segmentation algorithm achieves higher recognition rates than those achieved by a recent ontology-based method. Moreover, results indicate that our ontological reasoning technique is more effective than other unsupervised methods in mining semantic correlations. Indeed, our ontology encodes some important relationships between home infrastructure and activities that are hardly captured by fully automatic methods such as those based on Web mining. We have also performed experiments using Hidden Markov Model (HMM) as the probabilistic reasoning framework instead of MLN. Our MLN reasoner outperformed the HMM one, since the MLN model is capable of capturing complex semantic relationships and temporal aspects, while HMM only captures simple relationships.

A preliminary version of this work was presented in [13]. In this work, we extend our preliminary work with support for online activity recognition, thanks to a novel unsupervised segmentation algorithm. This extension provides support for time-critical applications, which are common in the healthcare domain [9]. We also conduct novel experiments showing that the accuracy achieved by the online algorithm is very close to the one of the offline algorithm.

The paper is structured as follows. We present related work and preliminary notions in Sections 2 and 3, respectively. In Section 4, we illustrate our activity model and the system overview. Section 5 presents ontological reasoning in POLARIS. In Section 6, we explain how we identify activity instance candidates, while in Section 7 we illustrate probabilistic reasoning in our framework. Experimental results are reported in Section 8. We discuss results and limitations in Section 9. Conclusions are reported in Section 10.

## 2 Related work

Several activity recognition systems rely on cameras and computer vision software [14]. Unfortunately, camera-based systems can raise serious privacy issues in smart-homes. Hence, in our work, we pursue sensor-based activity recognition. Methods for sensor-based activity recognition can be broadly classified in two categories: learning-based and specification-based methods [10].

Learning-based methods rely on a training set of sensor data, labeled with executed activities, and supervised learning algorithms to build the activities' model. Physical activity recognition systems are mainly based on data acquired from body-worn accelerometers [2], [15]. The same approach is extended with the use of environmental data acquired from other sensors (e.g., microphones) to recognize ADLs [16]. Observations regarding the user's surrounding environment (in particular, objects' use), possibly coupled with body-worn sensor data, are the basis of other activity recognition systems [17]. However, since training data is hard to acquire in realistic environments, systems relying on supervised learning are prone to serious scalability issues the more activities and the more context data are considered. Moreover, datasets of complex ADLs are strongly coupled to the environment in which they are acquired (i.e., the home environment and the sensors setup), and to the mode of execution of the specific individual. Hence, even if sophisticated methods for transfer learning have been proposed [18], the portability of activity datasets in different environments is an open issue [4]. In this work, we propose a method to recognize complex ADLs through semantic reasoning, even without the use of training data. However, when training data is available, we can exploit it to mine low-level dependencies between sensor events and performed activities. Those relationships are used by our probabilistic ontological reasoner to identify occurred activities.

Other machine learning approaches have been proposed to reduce the burden of dataset acquisition. Reinforcement learning [19] is a goal-oriented learning framework based on the concept of rewards assigned to an agent as a consequence of the actions it takes in the environment. It is not directly applicable to our case, since we aim at recognizing activities only by observing sensor data; i.e., we do not assume any feedback from the user or environment. To the best of our knowledge, it was never used for activity recognition. One-shot learning [20] is mainly used in computer vision and natural language understanding. It was also applied to gesture and scene recognition based on images and videos. To the best of our knowledge, it was never used for activity recognition based on sensor data. We believe that, due to the high variability of activity execution across different subjects and environments, one-shot learning is not a promising approach for activity recognition, at least when complex activities are considered as we do in this work.

Unsupervised learning algorithms build activity models relying on a training set of unlabeled sensor data. Some methods analyze textual descriptions of activities mined from the Web in order to obtain correlations among used objects and activities [21]. More recently, that approach has been extended exploiting visual cues extracted from the Web, such as images and videos [22]. In our work, we mine not only correlations, but also necessary conditions about sensor events that must be observed during the activity execution. Those conditions describe specific constraints derived from common sense, and are hardly captured by data mining techniques. Moreover, we derive correlations and necessary conditions considering the actual environment where activities are executed, while existing methods based on Web mining can only derive generic correlations.

Specification-based methods rely on knowledge-based definitions of the characteristics and semantics of complex activities. These are matched with available sensor data to recognize the current activity. Those definitions are usually expressed through logical axioms, rules, or description logics [23]. Ontological rea-

soning has also been proposed to perform dynamic segmentation of sensor data [24], [25] or to refine the output of supervised learning methods [26]. Further, probabilistic description logics have been used to recognize ADLs considering the variability of activity execution [27]. However, those works rely on rigid assumptions about the simpler constituents of activities. Hence, while the specification-based approach is effective for activities characterized by a few typical execution patterns, it is hardly scalable to the comprehensive specification of complex ADLs in different contexts. On the contrary, in this work we rely on general semantic relations among activities and smart-home infrastructure, which are fine-tuned to the current context.

In the literature, many hybrid approaches which combine data-driven and knowledge-based reasoning have been proposed to overcome many limitations of both worlds [17], [26], [28]. While we consider our method as hybrid (i.e., we combine semantic and probabilistic reasoning), differently from state-of-the-art hybrid approaches it does not have any data-driven component. Indeed, the classification is completely independent from data and it relies on semantic information derived from the ontology

Several works considered the challenging issue of segmenting temporal sequences of sensor data to accurately recognize the boundaries (i.e., start- and end-time) of activity instances in real time. However, very few works propose online and unsupervised segmentation methods. Most segmentation approaches do not operate in real-time [21], or they require a training set [29], [30]. Other ones operate in real-time, but they rely on rigid ontological and rule-based definitions of activities [31]. An unsupervised method that is close to our approach has been proposed by Ye et al. [28], where ontologies are used to derive semantic similarity between sensor events. This similarity is used to segment sensor data, obtaining sequential activities' patterns used to train a clustering model. With respect to that work, our method is totally independent from the data, it deals with interleaved activities, and it considers context information to segment sensor data.

## 3 Preliminaries

### 3.1 Description logics and formal ontologies

In computer science, description logics (DLs) [32] have emerged as the state-of-the-art formalism to represent *ontologies*. They enable the formal definition of concepts of a domain of interest, their properties, and the relationships among concepts. In this work, we use an ontology to formally define the semantics of activities, sensor events, and context data. Moreover, DLs support ontological reasoning, which allows to verify the consistency of the knowledge base, and to infer additional information from existing facts. The formalism of choice is typically OWL 2 [11]. A knowledge engineer can model the domain of interest by means of classes, instances, properties of instances, and relationships among instances. Several operators can be used to declare complex definitions based on simpler ones, including operators for conjunction, disjunction, negation, and universal and existential quantification. For instance, the activity PreparingHotMeal can be defined based on the definitions of PreparingMeal and PreparingColdMeal:

$$\text{PreparingHotMeal} \equiv \text{PreparingMeal} \sqcap \neg\text{PreparingColdMeal}$$

In this work, we use OWL 2 DL; i.e., a subset of OWL 2 having favorable computational properties. In particular, we exploit the following operators:

1) *Qualified cardinality restriction* restricts the class membership to those instances that are in a given relation with a minimum or maximum number of other instances of a given class. For instance, the following axiom states that PreparingHotMeal requires the use of at least one instrument to cook food:

$$\text{PreparingHotMeal} \sqsubseteq \text{Activity} \sqcap \geq 1\ \text{requiresUsageOf.CookingInstrument}$$

2) *Composition of properties.* OWL 2 supports a restricted form of property composition $\circ$. For instance, the following axiom states that if a person is in a given apartment, and she is executing a given activity, then that activity is executed in that apartment:

$$\text{executesAct}^{-} \circ \text{isInLocation} \sqsubseteq \text{actIsExecutedInLocation}$$

Note that executesAct$^{-}$ denotes the inverse of executesAct.

Formally, a DL knowledge base is composed by a pair $\langle \mathcal{T}, \mathcal{A} \rangle$. The *TBox* $\mathcal{T}$ constitutes the terminological part of the knowledge base. The TBox is composed of a set of axioms $C \sqsubseteq D$ or $P \sqsubseteq R$ (*inclusions*) and $C \equiv D$ or $P \equiv R$ (*equality*), where $C$ and $D$ are classes, and $P$ and $R$ are object properties. An axiom $C \sqsubseteq D$ is satisfied by an interpretation $\mathcal{I}$ when $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ satisfies a TBox $\mathcal{T}$ when $\mathcal{I}$ satisfies all the axioms of $\mathcal{T}$.

The *ABox* $\mathcal{A}$ is the assertional part of the knowledge base. The ABox is composed of a set of axioms of the form $x : C$ and $\langle x, y \rangle : R$, where $x$ and $y$ are instances, $C$ is a class in $\mathcal{T}$, and $R$ is an object property in $\mathcal{T}$. For instance, "mary : ElderlyPerson" denotes that Mary is an elderly person and "$\langle$ mary, apartment23 $\rangle$ : livesIn" represents that Mary lives in Apartment23. Axioms $x : C$ and $\langle x, y \rangle : P$ are satisfied by an interpretation $\mathcal{I}$ when $x^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\langle x^{\mathcal{I}}, y^{\mathcal{I}} \rangle \in P^{\mathcal{I}}$, respectively. An interpretation $\mathcal{I}$ satisfies an ABox $\mathcal{A}$ when $\mathcal{I}$ satisfies all the axioms of $\mathcal{A}$. An interpretation $\mathcal{I}$ that satisfies both the TBox $\mathcal{T}$ and the ABox $\mathcal{A}$ is called a *model* of $\langle \mathcal{T}, \mathcal{A} \rangle$. DLs support several reasoning tasks. In particular, we rely on the following ones:

- *Satisfiability:* a class $C$ is satisfiable with respect to a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\langle \mathcal{T}, \mathcal{A} \rangle$ such that $C^{\mathcal{I}}$ is non empty. We execute this reasoning task to check the consistency of our ontological model.
- *Property fillers retrieval:* given an object property op in $\mathcal{T}$ and an instance inst in $\mathcal{A}$, retrieving every instance inst$_j$ in $\mathcal{A}$ that is related to inst with respect to op; i.e., such that $\langle$ inst, inst$_j$ $\rangle$ : op belongs to $\mathcal{A}$. We execute this reasoning task to derive semantic correlations among activities and events.

### 3.2 Markov Logic with numerical constraints

The main idea of Markov Logic network is to allow rigid first-order-logic formulae to be "softened". The validity of a soft formula is evaluated according to the probability of being true with respect to a set of axioms describing reality. Each soft formula is associated to a weight that represents the confidence on the validity of the formula. The main task of MLN reasoning is to determine the most probable set of axioms representing reality that can be inferred based on the defined formulae and a set of observations (facts). Intuitively, formulae with higher weights will have higher influence in deriving these axioms.

More formally, a Markov Logic Network (MLN) $\mathcal{M}$ is a finite set of pairs $(F_i, w_i), 1 \leq i \leq n$, where each $F_i$ is an axiom in function-free first-order logic and $w_i \in \mathbb{R}$ [12]. Together with a

finite set of constants $C = \{c_1, ..., c_n\}$ it defines the *ground* MLN $\mathcal{M}_C$, i.e., the MLN in which axioms do not contain any free variables. This comprises one binary variable for each grounding of $F_i$ with weight $w_i$. Hence, a MLN defines a log-linear probability distribution over Herbrand interpretations

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\mathbf{x})\right) \quad (1)$$

where $n_i(\mathbf{x})$ is the number of satisfied groundings of $F_i$ in the possible world $\mathbf{x}$ and $Z$ is a normalization constant. Consequently, the weight $w_i$ associated to an axiom $F_i$ reflects the confidence about the truth value of $F_i$: the larger $w_i$, the larger the difference in log probability between a possible world that satisfies $F_i$ and one that does not.

In a previous work, we extended MLN with numerical constraints resulting in a formalism denoted $\mathsf{MLN_{NC}}$ [33]. In this paper, we use this extension to reason on the temporal domain of activities and sensor events. The constraints are predicates of the form $\theta \bowtie \psi$, where $\theta$ and $\psi$ denote variables, numerical constants, or algebraic expressions (that might contain elementary operators). In this context, the binary operator $\bowtie$ returns a truth value under a particular grounding.

***Definition 1*** *(*$\mathsf{MLN_{NC}}$*).* A numerical constraint NC is composed of numerical constants (e.g., elements of $\mathbb{N}$), variables, elementary operators or functions $(+, *, -, \div, \%, \sqrt{\ })$, standard relations $(>, <, =, \neq, \geq, \leq)$, and Boolean operators $(\wedge, \vee)$. An $\mathsf{MLN_{NC}}$ is a set of pairs $(\mathsf{FC}_i, w_i)$ where $\mathsf{FC}_i$ is a formula in first-order logic that may contain a NC and $w_i$ is a real number representing the weight of $\mathsf{FC}_i$.

***Example 1.*** Using $\mathsf{MLN_{NC}}$ enables to represent the axiom: the events "turning on the oven" and "opening the fridge" cannot belong to the same instance of meal preparation if their temporal distance is more than two hours:

$$\forall\ se_1,\ se_2,\ ai_1,\ ai_2,\ t_1,\ t_2 :$$
$$event(se_1,\ 'turnOnOven',\ t_1) \wedge$$
$$event(se_2,\ 'openFridge',\ t_2) \wedge$$
$$occursIn(se_1, ai_1) \wedge occursIn(se_2, ai_2) \wedge$$
$$\mathsf{NC}(t_1, t_2) \Rightarrow ai_1 \neq ai_2$$

**where** $\mathsf{NC}(t_1, t_2) = |t_1 - t_2| > 120$

Maximum a posteriori (MAP) inference is the task of finding the most probable world given some observations also referred to as evidence. Given the observed variables $E = e$, the MAP problem aims to find an assignment of all non-evidence (hidden) variables $X = x$ such that an interpretation $\mathbf{I}$ is a MAP state if and only if $\mathbf{I} = \underset{x}{\text{argmax}}\ P(X = x \mid E = e)$. Based on the MLN of sensor events and semantic axioms, we apply MAP inference to derive the most probable world; i.e., the most probable occurred activities. Note that Equation 1 allows us to reason with both *soft* and *hard* axioms. The former are probabilistic. The latter are deterministic: their weight is orders of magnitude larger than the one of probabilistic axioms; hence, they are necessarily satisfied in the most probable world.

## 4 Model and System Overview

We assume a smart-home instrumented with sensors to detect interactions with items and furniture, context conditions (e.g., temperature), and presence in certain locations. We denote by *activity class* an abstract activity (e.g., cooking and cleaning), and by *activity instance* the actual occurrence of an activity of a given class during a certain time period. For instance, during the execution of activity instance $ai_1$ (*preparing dinner*), the subject executes the operations $op_1$ (opening the silverware drawer) and $op_2$ (turning on the microwave oven). Supposing that sensors are available to detect these operations, $op_1$ and $op_2$ generate two sensor events $se_1$ (of type $et_1$) and $se_2$ (of type $et_2$), whose timestamp corresponds to the time of the respective operation.

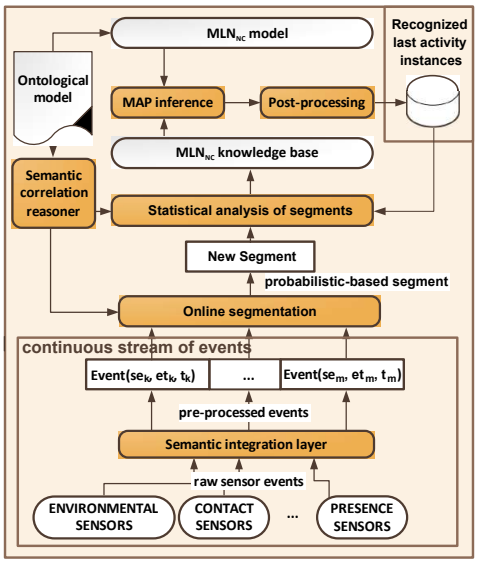



Fig. 1. System architecture of Online POLARIS.

Based on the observation of a set of timestamped sensor events, the goal of the activity recognition system is to reconstruct which activity instances generated those events. We achieve this goal by assigning each event $se_i$ to the activity instance that most probably generated it. This approach allows us to recognize interleaved activities, (e.g., the subject may temporarily interrupts the meal to take medicines). Depending on the application domain, POLARIS can perform activity recognition either in offline or online mode. Recognizing activities in an offline fashion means to analyze in batch mode a complete stream of sensor data acquired during a predetermined time period. For example, consider a system for cognitive health assessment of the elderly. That system should monitor the individual's behavior on the long-term. Hence, at the end of each day, the offline activity recognition algorithm may process all the sensor data acquired during that day. Offline ADL recognition is often preferred as it goes along with higher accuracy. However, other real-time monitoring applications, such as services that require intervention (e.g., reminders, emergency monitoring), require online recognition. That task is typically harder, since the recognition system must segment the continuous stream of sensor events on-the-fly in order to infer the most likely activity in nearly real-time and detect activity changes as they happen. In the following, we outline our overall framework and we explain its online version. The offline version of POLARIS is illustrated in [13].

The online mode of POLARIS relies on two specific layers, namely ONLINE SEGMENTATION and STATISTICAL ANALYSIS OF SEGMENTS (see Figure 1). The ONLINE SEGMENTATION method is in charge of segmenting the stream of sensor events in real-time. In particular, for each event in the stream, our segmentation method considers probabilistic and semantic conditions to decide whether to finalize

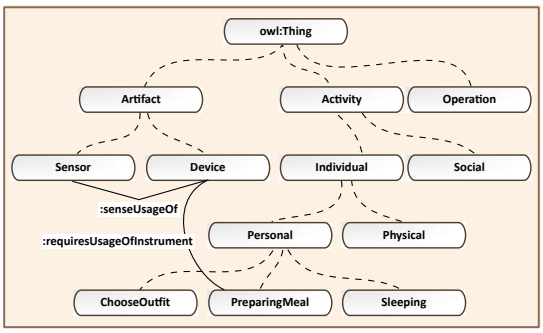



Fig. 2. Excerpt of our ontology. The dashed lines represent a *subClassOf* relation where the upper is the parent of the lower class. In addition, the individual classes have relations that describe dependencies.

a segment and initiate a new one. We call that operation a *split* decision. Our method aims at minimizing the number of generated segments, while ensuring that, with a high probability, the events belonging to a specific segment are labeled with the same activity class. More formally, given a temporal sequence of sensor events $\langle ev_1, ev_2, \ldots, ev_n, \ldots\rangle$, the role of the segmentation algorithm is to derive a set of segments:

$$\langle Segment(ev_1, \ldots, ev_l), \ldots, Segment(ev_m, \ldots, ev_n), \ldots\rangle,$$

where each segment $Segment(ev_j, ev_{j+1}, \ldots, ev_k)$ represents a set of consecutive and ordered sensor events from $ev_j$ to $ev_k$. Segments do not overlap and each sensor event is assigned to exactly one segment. As soon as a segment is finalized, it is immediately forwarded to the STATISTICAL ANALYSIS OF SEGMENTS layer, which analyzes the last $k$ generated segments to identify activity instance *candidates*. Finally, those *candidates* are refined by our $MLN_{NC}$ reasoner, thus inferring the most likely activities performed by the inhabitant.

## 5 ONTOLOGICAL REASONING

We adopt ontological reasoning to mine semantic correlations among event types and activity classes. Those correlations are considered by the $MLN_{NC}$ knowledge base in order to recognize activities. Further, we also mine the ontology to derive hard axioms used to enrich our $MLN_{NC}$ model with background knowledge about the activities' semantics. In the following, we introduce our ontological model[1] and subsequently we explain the above mentioned reasoning methods. For that purpose, we introduce a simple running example to illustrate our approach (see Example 2).

***Example 2.*** Suppose to monitor three activities in a smart-home: preparing hot meal, preparing cold meal, and preparing tea. The home contains: one silverware drawer, one stove, and one freezer, each equipped with a sensor to detect its usage. No training set of activities is available. How can we exploit semantic reasoning to recognize the activities?

### 5.1 Ontological model

We define the semantics of activities and operations in an OWL 2 ontology. In the following, we consider $\mathbf{A} = \{ac_1, ac_2, \ldots, ac_n\}$ as the set of activity classes. Further, an instance $ai_i$ of an activity class $ac_j \in \mathbf{A}$ represents the occurrence of $ac_j$ during a given timespan. An activity instance is associated to the operations that were executed to perform it, where the start and end time of instances of different activities can overlap.

Figure 2 illustrates an excerpt of our ontology, which describes a complete home environment. In addition, it also covers axioms for each activity class that describe dependencies and conditions. In particular, we express necessary conditions for a set of operations to be generated by an instance of that class, according to the activity semantics. For example, the operations generated by an instance of *preparing hot meal* must include an operation *using a cooking instrument*. In this context, the ontology also covers sensor classes and corresponding operations 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 *using a cooking instrument*. The ontology carefully describes these kinds of relations and, through ontological reasoning, we can derive certain constraints. Referring Example 2: "Since the stove is the only cooking instrument in the home, and a sensor is available to detect the usage of the stove, then each instance of *preparing hot meal* executed in the home must necessarily generate an event from that sensor".

In addition to activity and object correlations, we also take time and location dependencies into account. This includes constraints on the duration of the activity instance and the relation between an activity and a certain location. In the next section, we explain how we use ontological reasoning to infer these probabilistic dependencies among sensor event types and classes of executed activities; we denote them as *semantic correlations*.

It is important to note that logics underlying formal ontologies can be seen as a modeling language like ER is for relational databases. Similarly to ER design, the engineer needs domain knowledge. The deeper and wider the domain knowledge, the better and more general the resulting domain formalization. There is no fixed methodology for generating a perfect DL representation, and the process cannot be made automatic. However, like for ER design, there are some principles to help in the design process [1].

### 5.2 Semantic correlation reasoner

The specific objective of this reasoner is to compute the degree of correlation among sensor events and the activities performed in the home. As illustrated in the axioms below, in our ontology, artifacts are organized in a hierarchy. The class STOVE is a subclass of cooking instruments, used in the apartment to prepare hot meal or tea, where FREEZER is a DEVICE used to prepare hot or cold meal. SILVERWAREDRAWER belongs to FOODPREPFURNITURE and is used for the three activities. The instance {APT} represents the current apartment. For clarity, we represent the name of ontological instances within curly brackets.

$$\text{STOVE} \sqsubseteq \text{COOKINGINSTRUMENT} \sqcap \Big(\exists\, \text{USEDFOR}.((\text{PREPHOTMEAL} \sqcup \text{PREPTEA}) \sqcap (\exists\, \text{OCCURSIN}.\{\text{APT}\}))\Big).$$

$$\text{FREEZER} \sqsubseteq \text{DEVICE} \sqcap \Big(\exists\, \text{USEDFOR}.((\text{PREPHOTMEAL} \sqcup \text{PREPCOLDMEAL}) \sqcap (\exists\, \text{OCCURSIN}.\{\text{APT}\}))\Big).$$

$$\text{SILVERWAREDRAWER} \sqsubseteq \text{FOODPREPFURNITURE}.$$

1. Our ontology is publicly available: https://sensor.informatik.uni-mannheim.de/#results2017polaris

$$\text{FoodPrepFurniture} \sqsubseteq \text{Furniture} \sqcap \Big(\exists \text{usedFor}.((\text{PrepTea} \sqcup \text{PrepColdMeal} \sqcup \text{PrepHotMeal}) \sqcap (\exists \text{occursIn}.\{\text{apt}\}))\Big).$$

Based on the smart-home setup, we instantiate the ontology with the sensors and artifacts in the apartment, and we specify which activities we want to monitor.

***Example 3.*** The activities that we want to monitor are {ac_prep_cold_meal}, {ac_prep_hot_meal} and {ac_prep_tea}. They are instances representing the occurrences of PrepColdMeal, PrepHotMeal, and PrepTea, respectively. Lines 6-8 state that at most one instance of each activity type can be monitored at a time. Further, lines 9-11 represent that the {apt} contains exactly one cooking instrument, one silverware drawer, and a freezer:

$$\{\text{apt}\} = \text{Apartment} \tag{2}$$
$$\sqcap (\exists \text{monitAct}.(\{\text{ac\_prep\_cold\_meal}\})) \tag{3}$$
$$\sqcap (\exists \text{monitAct}.(\{\text{ac\_prep\_hot\_meal}\})) \tag{4}$$
$$\sqcap (\exists \text{monitAct}.(\{\text{ac\_prep\_tea}\})) \tag{5}$$
$$\sqcap (\leq 1 \text{monitAct}.\text{PrepColdMeal}) \tag{6}$$
$$\sqcap (\leq 1 \text{monitAct}.\text{PrepHotMeal}) \tag{7}$$
$$\sqcap (\leq 1 \text{monitAct}.\text{PrepTea}) \tag{8}$$
$$\sqcap (= 1 (\text{isIn})^{-}.\text{CookingInstrument}) \tag{9}$$
$$\sqcap (= 1 (\text{isIn})^{-}.\text{SilverwareDrawer}) \tag{10}$$
$$\sqcap (= 1 (\text{isIn})^{-}.\text{Freezer}). \tag{11}$$

Subsequently, we introduce an instance in the ontology for each artifact in the apartment:

$$\{\text{stove}\} \equiv \text{Stove} \sqcap \exists \text{isIn}.\{\text{apt}\}.$$
$$\{\text{freezer}\} \equiv \text{Freezer} \sqcap \exists \text{isIn}.\{\text{apt}\}.$$
$$\{\text{silverware\_drawer}\} \equiv \text{SilverwareDrawer} \sqcap \exists \text{isIn}.\{\text{apt}\}.$$

We also instantiate each sensor that occurs in our apartment:

$$\{\text{s\_stove}\} \equiv \text{PowerSensor} \sqcap (\exists \text{sensesUsageOf}.\{\text{stove}\}) \sqcap (\exists \text{producesEvent}.\{\text{et\_stove}\}).$$

$$\{\text{s\_silverware\_drawer}\} \equiv \text{ContactSensor} \sqcap (\exists \text{sensesUsageOf}.\{\text{silverware\_drawer}\}) \sqcap (\exists \text{producesEvent}.\{\text{et\_silverware\_drawer}\}).$$

$$\{\text{s\_freezer}\} \equiv \text{ContactSensor} \sqcap (\exists \text{sensesUsageOf}.\{\text{freezer}\}) \sqcap (\exists \text{producesEvent}.\{\text{et\_freezer}\}).$$

According to the introduced axioms, {s_stove} is an instance of PowerSensor which senses the usage of {stove} and produces an event of type {et_stove}. Similarly, the last two axioms define sensors and events for the silverware drawer and the freezer, respectively.

We exploit the property composition operator to infer the semantic correlations between sensor events and activity types. In particular, we use the following axiom, which states that: "if an event of type *et* is produced by a sensor that detects the usage of an artifact possibly used for an activity of class *ac*, then *et* is a *predictive sensor event type* for *ac*":

$$\text{producesEvent}^{-} \circ \text{sensesUsageOf} \circ \text{usedFor} \rightarrow \text{predictiveSensorEventFor}$$

Then, we perform ontological reasoning to infer the fillers of property predictiveSensorEventFor, and use them to compute semantic correlations.

***Example 4.*** Considering all of the introduced axioms, the OWL 2 reasoner infers that:

- {et_stove} *is a predictive sensor event type for* {ac_prep_hot_meal} *and* {ac_prep_tea}.
- {et_silverware_drawer} *is a predictive sensor event type for* {ac_prep_hot_meal},{ac_prep_cold_meal} *and* {ac_prep_tea}.
- {et_freezer} *is a predictive sensor event type for* {ac_prep_hot_meal} *and* {ac_prep_cold_ meal}.

We denote as $predAct(et)$ as the set of activities for which $et$ is a predictive event type. We represent semantic correlations using a *prior probability matrix* (*PPM*). The rows correspond to the activity classes, while the columns to the sensor event types. Hence, $PPM(ac, et)$ stores the probability of an event of type $et$ being generated by an activity of class $ac$. Hence, given $et$, we have that $PPM(ac, et)$ is a probability distribution over all $ac$ values:

$$\forall\, et \in \mathbf{E} \sum_{ac \in \mathbf{A}} PPM(ac, et) = 1. \tag{12}$$

To enforce property (12), we set the values of the prior probability matrix *PPM* for each combination of event type $et$ and activity class $ac$ in the following way. We consider two cases. If $et$ is predictive of at least one activity class, we compute $et$'s correlations using the following formula:

$$PPM(ac, et) = \begin{cases} \frac{1}{|predAct(et)|} & \text{if } ac \in predAct(et) \\ 0 & \text{otherwise} \end{cases}$$

Otherwise, having no information about the associations between $et$ and activity classes, we uniformly distribute its correlation values across all possible activity classes. It is easy to verify that in both cases property (12) is enforced. The prior probability matrix resulting from our running example is shown in Table 1.

TABLE 1
Prior probability matrix of our running example.

| | {et_stove} | {et_silverware_drawer} | {et_freezer} |
|---|---|---|---|
| {ac_prep_hot_meal} | 0.5 | $0.\overline{33}$ | 0.5 |
| {ac_prep_cold_meal} | 0.0 | $0.\overline{33}$ | 0.5 |
| {ac_prep_tea} | 0.5 | $0.\overline{33}$ | 0.0 |

The output of the semantic correlation reasoner layer (i.e., the PPM) is directly considered in the statistical analysis of segments layer that in turn generates the $MLN_{NC}$ knowledge base.

### 5.3 Deriving necessary sensor observations

In contrast to the semantic correlation reasoner, which is essentially used to build the $MLN_{NC}$ knowledge base, the following part focuses on using hard axioms extracted from the ontology to enrich our $MLN_{NC}$ model. Our ontology includes a property requiresUsageOfArtifact, which associates artifacts in the apartment with activities for which they are necessary.

***Example 5.*** Continuing our running example, the axiom below defines PrepHotMeal as a subclass of PrepareMeal that requires the usage of a cooking instrument:

$$\text{PrepHotMeal} \sqsubseteq \text{PrepareMeal} \sqcap \exists \text{requiresUsageOfArtifact}.(\text{CookingInstrument} \sqcap (\exists \text{isIn}.\{\text{apt}\})).$$

Subsequently, we infer which sensor events must necessarily be observed during the execution of an activity. The following axiom states that: "if an event of type *et* is produced by a sensor that detects the usage of an artifact required for executing an activity of class *ac*, then *et* is a *necessary sensor event type* for each activity instance of class *ac*".

$$\text{producesEvent}^- \circ \text{sensesUsageOf} \circ \text{requiresUsageOf}^- \rightarrow \text{necessaryEventFor}.$$

Then, we infer the fillers of property necessaryEventFor through ontological reasoning, translate them in $MLN_{NC}$ axioms, and add them, finally, to the $MLN_{NC}$ model.

***Example 6.*** Given the introduced axioms, in this case the OWL 2 reasoner infers that {et_stove} is a necessary sensor event type for {ac_prep_hot_meal}. Indeed, et_stove is produced by usage of stove, which is the only instance of CookingInstrument available in the home.

## 6 Identifying Activity Instance Candidates

Besides semantic correlations and ontological axioms, the input of our $MLN_{NC}$ probabilistic reasoner is a set of activity instance *candidates*: initial hypothesis about type, start- and end-time of the activity instances actually performed by the inhabitant. POLARIS infers those *candidates* from the stream of sensor events by using an heuristic algorithm. In the following, we describe how we derive activity instance *candidates* in the online version of POLARIS. The description of the algorithm used for the offline mode of POLARIS can be found in [13].

First, the stream of sensor events is continuously segmented considering several probabilistic and semantic conditions that we call *aspects*. Each aspect represents an indicator that the inhabitant possibly changed his/her current activity. These aspects aim at generating segments which cover at most one activity instance: we prefer to span an activity instance on multiple segments instead of trying to build a segment that perfectly fits an activity instance, as this would also increase the risk of including unrelated sensor events. This also allows to handle interleaved activities.

Finally, the statistical analysis of segments algorithm analyzes the last $k$ generated segments in order to derive activity instance *candidates* by considering the semantic correlations obtained from the ontology.

### 6.1 Online segmentation

The *Online Segmentation* algorithm considers five aspects: *Object interaction*, *Change of context*, *Consistency likelihood*, *Time leap*, and *Change of location*. Whenever a new sensor event $ev_{new}$ is detected, all those aspects are evaluated. If at least one aspect determines sufficient conditions to perform segmentation, the current segment is finalized and a new one (with $ev_{new}$ as the first element) is initialized. In the following, we outline the mentioned aspects.

ASP1) For each object, POLARIS keeps track of its usage status: *in use* or *not in use*. The usage status of each object is automatically updated according to the events in the stream. The *Object interaction* aspect finalizes a segment as soon as POLARIS detects that the user stopped interacting with all the objects in the home. For instance, suppose that the type of the current event $ev_{new}$ is "turning off the stove". If, at the same time, the subject is not actively using any other instrument, the current segment is finalized; indeed, the current activity is likely terminated. On the other hand, if the subject is using other objects at that time (e.g., the oven), the segment is not finalized.

ASP2) The *Change of context* aspect considers our ontological model to verify whether the new event in the stream ($ev_{new}$) is correlated with the last event of the current segment ($ev_{last}$). In this context, only sensor events related to an interaction are considered; e.g., temperature or presence sensor events are disregarded. Formally, we define

$$possAct(ev(se, et, t)) = \{ac \in \mathbf{A} : PPM(ac, et) > 0\}$$

as the set of possible activities for an event $ev$ given the semantic correlations. If $possAct(ev_{last}) \cap possAct(ev_{new}) = \emptyset$, the aspect derives that $ev_{new}$ cannot be labeled with the same activity class of $ev_{last}$, and thus the current segment is finalized.

ASP3) The *Consistency likelihood* aspect keeps track of the probability that the current segment includes events mostly labeled with the same activity class. Differently from ASP2, in this aspect we consider the whole set of the segment's events. In particular, we consider the semantic correlation among those events and possible activities, and we finalize the segment if the introduction of the new event $ev_{new}$ determines an abrupt shift in the likelihood of the segment, computed by the following formula:

$$L(S) = \max_{ac_i \in \mathbf{A}} \frac{\sum_{ev_j(se,et,t) \in S} PPM(ac_i, et)}{|S|},$$

where $PPM(ac_i, et)$ is the semantic correlation between activity $ac_i$ and event type $et$. If the fluctuation of $L(S)$ due to the introduction of $ev_{new}$ in $S$ exceeds an experimentally chosen threshold $\sigma$, the current segment is finalized.

ASP4) The *Time leap* aspect considers the time distance between consecutive events. If no new event is observed after the most recent event $ev_{last}$ according to a time threshold $\delta$, the current segment is finalized. The value of $\delta$ is automatically calibrated based on the stream of sensor events. In particular, we continuously keep track of the third quartile value $q$ of the temporal distances between consecutive sensor events. The value of $\delta$ is automatically updated as $2q$ whenever a new segment is finalized. Therefore, the *Time leap* aspect is not considered for the very first segment.

ASP5) The *Change of location* aspect relies on the fact that most ADLs are performed in a specific location. For that reason, we finalize the segment when the individual moves from a room to a different one.

The combination of those aspects aims to group sensor events that likely belong to the same activity instance. However, when the duration of an activity instance is particularly long (e.g., cooking for an hour), using only those aspects could generate segments which span for a long period. Since our goal is real-time activity recognition (i.e., detecting as soon as possible the current activity), our $MLN_{NC}$ is triggered every $n$ minutes even if a segment is not finalized. The threshold $n$ is chosen according to the specific needs of the application.

### 6.2 Statistical analysis of segments

The goal of the statistical analysis of segments module is to generate a set of activity instance candidates by analyzing the $k$ most recent segments. Initially, each segment is considered as a candidate: we analyze the PPM for each event in the segment to infer the most likely activity of the whole segment. Then, we merge those candidates (and the corresponding segments) which are temporally close and that are associated with the same activity class. This operation allows us to consider interleaved activities. The output is a set of activity instance candidates.

## 7 Probabilistic reasoning

As explained before, the method presented in Section 6.2 provides an initial hypothesis about the class and the temporal boundaries of activity instance candidates. The goal of probabilistic reasoning in POLARIS is to refine those initial hypothesis exploiting several kinds of domain knowledge expressed through our $MLN_{NC}$ model. In the following, we illustrate our model and probabilistic reasoning methods.

### 7.1 MLN modeling

Semantic correlations are modeled through predicates *PriorProb*, *Event*, and *Instance*. The *PriorProb* predicate represents correlations among sensor events and activities:

$$*PriorProb(SensorEvent, ActivInstance, ActivClass, p)$$

Hence, it describes the probability $p$ that a given sensor event corresponds to a given activity instance of an activity class. The probability relies on the semantic correlation between the event type and the activity class (PPM), and also depends on the temporal distance between the sensor event and the boundaries of the activity instance.

Formally, given an activity instance $ai$ of class $ac$ with start time $t_{st}$ and end time $t_{ed}$, and a sensor event $se$ of type $et$ and timestamp $t$, the probability $p$ of $*PriorProb(se, ai, ac, p)$ is computed by the following function:

$$p = \begin{cases} PPM(ac, et) & \text{if } t_{st}\text{-}MaxDelay_{ac} \leq t \leq t_{ed}\text{+}MaxDelay_{ac} \\ 0 & \text{otherwise} \end{cases}$$

Each sensor event is represented by an instance of the predicate *Event*, which represents the identifier, its type, and its timestamp:

$$*Event(SensorEvent, EventType, Timestamp)$$

Activity instance candidates are represented by the predicate *Instance* which models the relation between the activity instance, its start time, and end time:

$$*Instance(ActivInstance, STime, ETime)$$

The instantiated predicates, derived from the activity instances and the recorded sensor events, are added as facts to our $MLN_{NC}$ knowledge base.

### 7.2 Hidden predicates and domain constraints

Beside the observed predicates, the model also comprises a set of hidden predicates, which can be considered as our target classes: *Prediction*, *OccursIn*, and *InstanceClass*. The predicate *Prediction* represents the predicted assignment of a sensor event to an activity instance of a given class:

$$Prediction(SensorEvent, ActivInstance, ActivClass)$$

In addition, the other two predicates are used to express domain constraints about the consistency of inferred activity instances:

$$OccursIn(SensorEvent, ActivInstance)$$
$$InstanceClass(ActivInstance, ActivClass)$$

In particular, the following domain constraint states that each sensor event occurs in exactly one activity instance:

$$|ai|OccursIn(se, ai) = 1,$$

while the following one states that each activity instance belongs to exactly one activity type:

$$|ac|InstanceClass(ai, ac) = 1.$$

### 7.3 Semantic correlation rules

The relations between the observed and hidden predicates are modeled by probabilistic axioms. As illustrated in Figure 3, the hidden predicate *Prediction* is derived from *PriorProb*:

$$conf : *PriorProb(se, ai, ac, conf) \Rightarrow Prediction(se, ai, ac).$$

Thus, the confidence value describes the probability that a sensor event is assigned to an activity instance of a given class. In turn, the remaining hidden predicates are derived from the hidden *Prediction* predicate. The corresponding probabilistic axioms are the following:

$$Prediction(se, ai, ac) \Rightarrow OccursIn(se, ai),$$
$$Prediction(se, ai, ac) \Rightarrow InstanceClass(ai, ac).$$

Note that the above rules are subject to the domain constraints introduced before.

### 7.4 Knowledge-based constraints

Knowledge-based constraints enable us to express conditions about the occurrence (or non-occurrence) of sensor events of a given type during the occurrence of an activity instance.

***Example 7.*** The constraint "each activity instance of type *preparing hot meal* must be associated to an event of type *UseStove*" is logically expressed by the rule:

$$InstanceClass(ai, \text{“}PreparingHotMeal\text{”}) \Rightarrow \exists\ se, t :$$
$$OccursIn(se, ai) \wedge *Event(se, \text{“}UseStove\text{”}, t).$$

Knowledge-based constraints are automatically derived from the fillers of the necessaryEventFor OWL 2 property obtained from ontological reasoning as already mentioned.

### 7.5 Temporal constraints

We model $MLN_{NC}$ temporal constraints regarding the duration and the distance of events or activities. Temporal thresholds, represented below as $\Delta$ and $\Delta'$, are derived from the ontology and hence they require human knowledge-engineering effort based on common-sense. Those thresholds could also be tuned over time through active learning and periodic ontology revisions. Moreover, when training data is available, the value of those thresholds can be mined from the data. We consider two kinds of temporal constraints:

1) *Temporally close events (e.g., whose temporal distance is below $\Delta$ seconds) likely belong to the same activity instance*. We express this constraint through the following axioms:

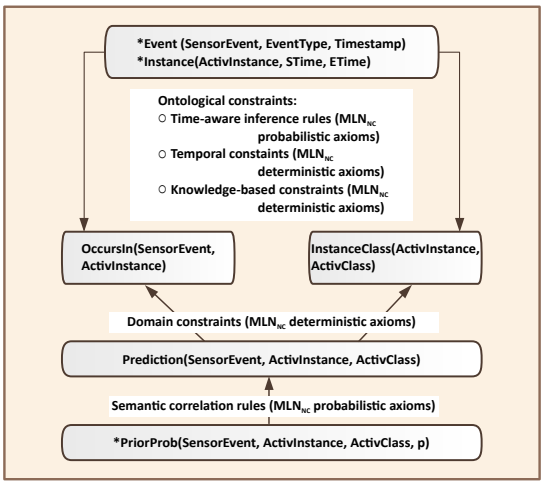



Fig. 3. Probabilistic activity recognition framework. The arrows indicate the relations and dependencies between the depicted observed and hidden predicates.

$$\forall\, t_1,\, t_2\, :\, (|t_1 - t_2| < \Delta) \;\Rightarrow\; tClose(t_1,\, t_2)$$

$$w\ Event(se_1,\, et_1,\, t_1)\ \wedge\ Event(se_2,\, et_2,\, t_2)\ \wedge$$
$$tClose(t_1,\, t_2) \wedge OccursIn(se_1,\, ai)\ \Rightarrow\ OccursIn(se_2,\, ai)$$

The latter is a soft axiom whose weight $w$ is chosen experimentally.

2) *Constraints on typical duration of each activity (e.g., "showering cannot last more than $\Delta'$ minutes").* We express these constraints either through probabilistic or deterministic axioms, according to the characteristics of the considered activity. Indeed, the variance of the duration of certain activities (e.g., showering) is relatively small, while it is larger for other activities (e.g., preparing dinner). The duration of the former is modeled with deterministic axioms where probabilistic ones are used for the latter. The axioms below state that an instance of "showering" cannot last more than $\Delta'$ minutes:

$$\forall\, t_1,\, t_2\, :\, (|t_1 - t_2| < \Delta')\ \Rightarrow\ tclose\_showering(t_1,\, t_2)$$

$$InstanceClass(ai, \text{“}Showering\text{”})\ \wedge\ OccursIn(se_1, ai) \wedge$$
$$OccursIn(se_2, ai)\ \wedge\ Event(se_1, et_1, t_1) \wedge$$
$$Event(se_2, et_2, t_2)\ \Rightarrow\ tclose\_showering(t_1,\, t_2)$$

### 7.6 Time-aware inference rules

Finally, as explained before, the semantics of some simple activities are naturally expressed in our ontology based on the typical actions composing them. Hence, we apply rules that express the relation of specific operations derived from sensor events in context of time. Consider the following example:

***Example 8.*** A typical pattern of operations for watering plants consists in (1) "getting water" and (2) "moving to the plants" shortly after. We express this activity inference pattern through the $MLN_{NC}$ axioms below:

$$Event(se_1,\, \text{“}water\_sensor\text{”},\, t_1)$$
$$\wedge\, Event(se_2,\, \text{“}plant\_presence\_sensor\text{”},\, t_2)\ \wedge\ t_1 < t_2$$
$$\wedge\, tclose\_waterplants(t_1,\, t_2)\ \Rightarrow\ \exists\, ai\ :$$
$$InstanceClass(ai,\, \text{“}WaterPlants\text{”})$$
$$\wedge\, occursIn(se_1,\, ai)\ \wedge\ occursIn(se_2,\, ai).$$

### 7.7 Inference of activity classes and instance boundaries

In order to reconstruct the relations of activity instances, their class, and the corresponding sensor events, we execute MAP INFERENCE on the presented $MLN_{NC}$ model by considering the introduced and generated $MLN_{NC}$ knowledge base. The result is a set of *OccursIn* and *InstanceClass* predicates. The former maps a sensor events onto the most probable corresponding activity instance where the latter assigns the most likely activity class to an activity instance. These (hidden) predicates are post-processed in order to detect the class and temporal boundaries of each activity instance $ai$:

$$AClass(ai) = ac : \exists InstanceClass(ai, ac),$$
$$STime(ai) = \min\{t\ :\ \exists Event(se, et, t) \wedge OccursIn(se, ai)\},$$
$$ETime(ai) = \max\{t\ :\ \exists Event(se, et, t) \wedge OccursIn(se, ai)\}.$$

In this context, $AClass(ai)$ represents the activity class of $ai$, while $STime(ai)$ and $ETime(ai)$ respectively the start- and end-time. Computing the start and end time of activity instances by the $MLN_{NC}$ resolver would be unnecessarily complicated, hence, they are computed in a post-processing phase. This post-processing step also varies slightly depending on the recognition mode. Indeed, in online mode the recognition never stops. The overall result is a sequence of activities that most likely caused the recorded sensor events.

## 8 Experimental evaluation

In the following, we present our experimental setup and results. We show results for both offline (Offline POLARIS) [13] and online (Online POLARIS) versions of our system. In order to facilitate the reproduction of our results we make publicly available a REST API for the $MLN_{NC}$ solver including a web interface, the MLN model, and the ontology[2].

### 8.1 The CASAS and SmartFABER datasets

In order to evaluate our method, we consider the well-known dataset of Cook et al. [34], named *CASAS*, and the dataset presented in [1], called *SmartFABER*. Both datasets include interleaved activities in a smart-home environment.

The CASAS dataset covers interleaved ADLs of twenty-one subjects acquired in a smart-home laboratory. Sensors collected data about movement, room temperature, use of water, and interaction with certain objects and doors. For that purpose, 70 sensors were used in total where eight activities were observed: *fill medication dispenser* ($ac_1$), *watch DVD* ($ac_2$), *water plants* ($ac_3$), *answer the phone* ($ac_4$), *prepare birthday card* ($ac_5$), *prepare soup* ($ac_6$), *clean* ($ac_7$), and *choose outfit* ($ac_8$). The order and expenditure of time were up to the subject and it was allowed to perform the activities in parallel. During the data collection only one single person was present in the smart-home. With our method, only 25 out of 70 sensors are required. Indeed, the semantic correlation reasoner excludes the remaining 45 (mostly movement sensors), since they have no significant semantic correlation with the considered activities.

The SmartFABER dataset has been acquired as part of the SECURE interdisciplinary project [17]. Sensor data were acquired from the apartment of an elderly woman diagnosed with Mild Cognitive Impairment. Different environmental sensors (magnetic, motion, and temperature) were used to monitor three ADLs for

2. https://sensor.informatik.uni-mannheim.de/#results2017polaris

55 days: *taking medicines* ($ac_9$), *cooking* ($ac_{10}$), and *eating*. The remaining activities were labeled as *others* ($ac_{11}$). Totally, 11 sensors were deployed. Our semantic correlation reasoner discards two sensors of these as they have no significant semantic correlation with the considered activities. Unfortunately, we were unable to recognize *eating* because it is only characterized by a single presence sensor close to the table. Indeed, this sensor is also activated in context of all other activities. Hence, our semantic correlation reasoner did not find any sensor that is significantly correlated with *eating*. Therefore, we decided to exclude that activity from the evaluation. On the other side, we are able to recognize *others* ($ac_{11}$), which was not considered in [17]. Compared to CASAS, this dataset was acquired in a fully naturalistic environment. Due to the cognitive decline of the subject, activities were performed in many different and sometimes unexpected ways. Besides, the acquired data is also affected by noise due to various technical issues encountered during data acquisition [1]. Hence, the recognition of ADLs in this scenario is challenging, even if the number of considered activities is small.

### 8.2 Segmentation effectiveness

#### 8.2.1 Segmentation Evaluation Metrics

Compared to our previous work [13], the additional challenge introduced by online recognition consists in the need for segmenting the continuous stream of sensor events on-the-fly. In order to obtain good recognition rates, the segmentation quality is a crucial aspect. An optimal segmentation strategy would map each segment to exactly one activity instance. In order to evaluate the effectiveness of segmentation, we use two metrics, called *purity* and *deviation of segments* (DS for brevity), respectively. A segment $S$ is perfectly pure (i.e., its purity value is equal to 1) when all of its events $ev_i \in S$ are labeled with the same activity class. The formula to compute the purity of a segment $S$ is given below:

$$purity(S) = \max_{ac \in \mathbf{A}} \sum_{ev_i \in S} \frac{\mathbb{1}[ev_i \text{ is labeled } ac]}{|S|} \tag{13}$$

We compute the overall purity of a set of segments $\mathbf{S}$ as the average of $purity(S)$ $\forall S \in \mathbf{S}$, weighted according to the size of each segment. The second metric, DS, is computed as the root mean square of the segmentation error in terms of the number of inferred segments. Formally, considering a sequence of sensor events $E = \langle Event(se_1, et_1, t_1), \ldots, Event(se_n, et_n, t_n) \rangle$, we denote $S_{E,A}$ the set of segments for $E$ predicted by a segmentation algorithm $A$, and we denote $\overline{S}_E$ the exact set of segments of $E$. The segmentation error $\epsilon(S_{E,A}, \overline{S}_E)$ is computed as the modulus of $|S_{E,A}| - |\overline{S}_E|$. Hence, given a set of sequences of sensor events $\mathcal{E} = \{E_1, E_2, \ldots, E_j\}$, we compute the DS of $A$ by the following formula:

$$DS(\mathcal{E}, A) = \sqrt{\sum_{E \in \mathcal{E}} \frac{\epsilon(S_{E,A}, \overline{S}_E)}{|\mathcal{E}|}}$$

#### 8.2.2 Evaluating the impact of segmentation

We evaluate our segmentation algorithm in terms of the two metrics described above as well as in terms of the resulting recognition rate ($F_1$ score). The evaluation is performed considering two other segmentation approaches as baselines. The first is a widely used method in mobile sensor data processing [35] that we call *Naive Segmentation*. Each segment is taken as a sliding window covering $w$ sensor events with overlap factor $o$. This method does not exploit any information from the ontology. In order to compare with a similar knowledge-based technique for online activity segmentation, we chose one of the very few proposals in this category, namely, the algorithm described in USMART [28], that proved to be particularly effective in the literature. As we mentioned in Section 2, to the best of our knowledge, this is the unsupervised and real-time segmentation method closest to our approach. USMART uses an ontology to compute semantic similarity between each pair of consecutive sensor events. If two consecutive events are similar, they are considered part of the same segment. Otherwise, they are considered part of different segments. The similarity is computed using the WordNet lexical database. We have empirically determined that the best parameters (in terms of resulting recognition rate) of the *Naive Segmentation* are $w = 6$ and $o = 50\%$ for the CASAS dataset, while they are $w = 4$ and $o = 50\%$ for the SmartFABER dataset. Regarding the semantic similarity threshold for USMART's segmentation method, we empirically chose 0.75 for the CASAS dataset and 0.625 for the SmartFABER dataset.

Table 2 shows the effectiveness of our segmentation approach on both datasets, compared with the considered baselines. Online POLARIS outperforms both the naive approach and the segmentation method proposed in USMART. The superiority of our approach over USMART, in terms of $F_1$, is probably due to the fact that we exploit rich semantic knowledge of relations between sensor events and activities that are not captured by the mere ontological similarity between pairs of sensor events. Moreover, our segmentation algorithm exploits several semantic and temporal heuristics to continuously segment the stream of sensor data. Even if Online POLARIS performs significantly better than the naive approach, from the results it emerges that the naive method sometimes reaches recognition rates close to ours. However, as observed above, it generates a much higher number of segments; this leads to triggering the MLN reasoner much more often with respect to our solution, thus slowing down the classification process, which is intended to run in real-time. Finally, the naive approach does not consider time, but separates segments only based on event sequences. Hence, two events within the same segment may be temporally distant. This behavior may cause a significant delay in activity recognition, since a segment is classified only when it is complete. Hence the naive segmentation approach is less suitable for real-time applications.

Figure 4 shows how *purity*, DS, and overall $F_1$ score change by varying the segmentation algorithm. In addition to the above mentioned baselines, we also evaluated different combinations of the aspects of our segmentation technique, which we introduced in Section 6.1; we reported the results only for those combinations which achieved acceptable recognition rates. Even if POLARIS (i.e., where we use all the five aspects) does not achieve the lowest DS value, it achieves a good *purity* and the best recognition results with respect to the considered segmentation techniques on both datasets. Considering subsets of Online POLARIS segmentation aspects leads in general to worse recognition rates. However, we observed that ASP2 alone leads to a very high F1 score on the SmartFABER dataset. As explained in Section 6.1, this aspect captures the change of context: a split decision occurs when an event is not correlated according to the ontology to its previous event. Considering that this dataset has few activities that are very different among them, this aspect alone already allows to reach good results. However, as it emerges for the CASAS dataset, when

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930050, IEEE Transactions on Knowledge and Data Engineering



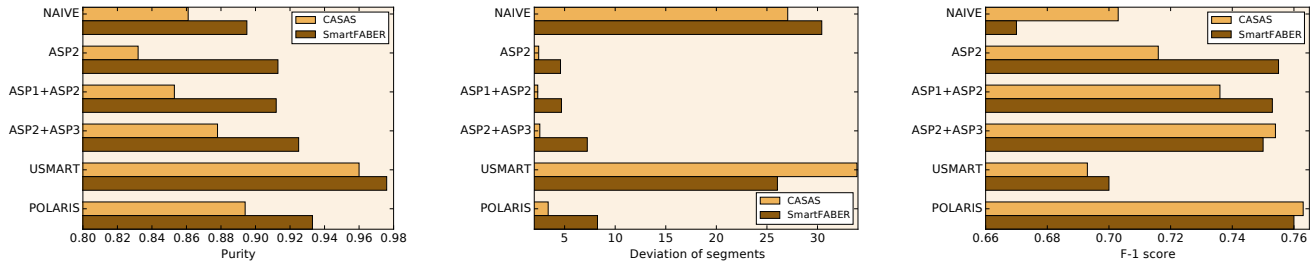
TABLE 2
Comparison of different segmentation strategies ($F_1$ score)

| Class | Naive Segmentation | USMART [28] Segmentation | POLARIS Segmentation |
|---|---|---|---|
| $ac_1$ | 0.70 | **0.83** | 0.74 |
| $ac_2$ | 0.84 | **0.87** | 0.86 |
| $ac_3$ | 0.57 | 0.51 | **0.62** |
| $ac_4$ | 0.56 | 0.44 | **0.74** |
| $ac_5$ | 0.92 | 0.87 | **0.93** |
| $ac_6$ | 0.83 | 0.62 | **0.88** |
| $ac_7$ | 0.51 | 0.54 | **0.56** |
| $ac_8$ | 0.69 | **0.86** | 0.77 |
| *avg.* | 0.70 | 0.69 | **0.76** |

(a) CASAS dataset

| Class | Naive Segmentation | USMART [28] Segmentation | POLARIS Segmentation |
|---|---|---|---|
| $ac_9$ | 0.73 | **0.83** | 0.81 |
| $ac_{10}$ | 0.65 | 0.58 | **0.76** |
| $ac_{11}$ | 0.63 | 0.70 | **0.71** |
| *avg.* | 0.67 | 0.70 | **0.76** |

(b) SmartFABER dataset




(a) Segments average *purity* (b) Deviation of segments (DS) (c) ADLs recognition's $F_1$

Fig. 4. How *purity*, deviation of segments (DS) and $F_1$ vary by changing the online segmentation technique.

the number of activities is higher, ASP2 alone is not sufficient for an effective segmentation.

Inspecting the results of naive approach and USMART, it emerges that they reach a good *purity*, but they are affected by a high DS value. This is due to the fact that both approaches produce a high number of segments, negatively impacting recognition results. The very high purity of USMART is due to the fact that this technique generates very small segments containing events generated by the same activity instance.

Finally, purity and DS metrics are generally higher on the SmartFABER dataset. On the one hand, the limited number of activities that compose this dataset likely increases the probability of having segments with events labeled with the same activity. On the other hand, due to the high variability of execution in this dataset, our segmentation approach generates more segments.

### 8.3 Classification results

In order to evaluate the Online POLARIS method, we compared its recognition rate for both datasets with a few other methods. In the following, we present these methods and comment the results reported in Table 3.

Despite our contribution is an unsupervised method avoiding the recognised problems of supervised ones, we implemented and evaluated several supervised methods to elect one as a reference for recognition rate values. Indeed, since supervised methods are considered effective for this type of problems, achieving recognition rates close to the best of these algorithm is a desirable property. For all algorithms, we adopted the state-of-the-art feature extraction technique presented in [36], which is particularly suitable for real-time activity recognition. As Figure 5 shows, Random Forest and MLP emerge but Random Forest was preferred to MLP for computational efficiency reasons.

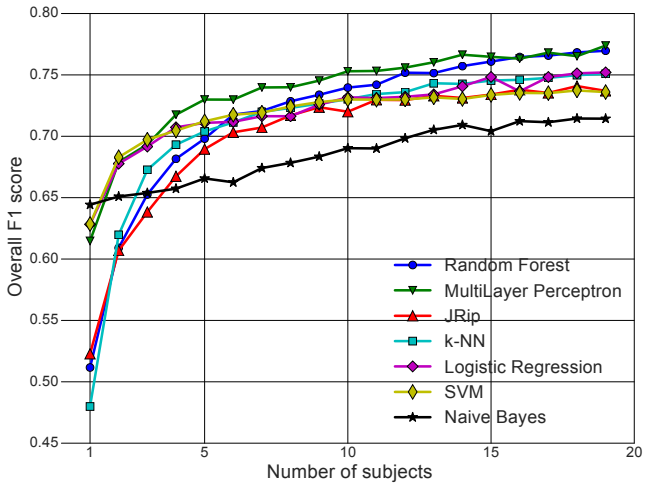



Fig. 5. CASAS dataset: How the number of persons used to train the recognition model (x-axis) affects the overall F1 score (y-axis) of different classifiers.

From Table 3, Online POLARIS achieves almost the same recognition rate than Random Forest for the CASAS dataset, while it gives an average recognition rate 4% lower on the SmartFABER dataset. This is probably due to the fact that this dataset consists of activities performed by a single subject in a single home. Hence, with that dataset we could not evaluate the capability of the classifier to generalize on different subjects or environments. On the other hand, for the CASAS dataset we used a leave-one-subject-out cross validation, thus evaluating the generalization capability of the reasoner.

TABLE 3
Recognition rate of Online POLARIS compared with the baselines

| Class | Random Forest [36] | Semantic HMM | Offline POLARIS | Web POLARIS | Online POLARIS |
|---|---|---|---|---|---|
| $ac_1$ | 0.74 | 0.72 | **0.85** | 0.68 | 0.74 |
| $ac_2$ | 0.87 | 0.85 | 0.81 | 0.79 | **0.86** |
| $ac_3$ | 0.57 | 0.65 | **0.72** | 0.50 | 0.62 |
| $ac_4$ | 0.72 | 0.27 | 0.72 | 0.23 | **0.74** |
| $ac_5$ | 0.91 | 0.84 | 0.81 | 0.80 | **0.93** |
| $ac_6$ | 0.85 | **0.89** | 0.88 | 0.69 | 0.88 |
| $ac_7$ | 0.67 | **0.64** | 0.57 | 0.52 | 0.56 |
| $ac_8$ | 0.84 | 0.81 | **0.88** | 0.69 | 0.77 |
| *avg.* | 0.77 | 0.71 | **0.78** | 0.61 | 0.76 |

(a) CASAS dataset

| Class | Random Forest [36] | Semantic HMM | Offline POLARIS | Web POLARIS | Online POLARIS |
|---|---|---|---|---|---|
| $ac_9$ | 0.91 | **0.86** | 0.83 | 0.82 | 0.81 |
| $ac_{10}$ | 0.77 | 0.68 | 0.75 | 0.66 | **0.76** |
| $ac_{11}$ | 0.73 | **0.71** | 0.70 | 0.50 | **0.71** |
| *avg.* | 0.80 | 0.74 | 0.76 | 0.66 | **0.76** |

(b) SmartFABER dataset

Next, we compare Online POLARIS with its offline version. Despite we would expect a cost in terms of accuracy for the increased utility of an online algorithm, the results show that the accuracy of Online POLARIS is similar to the one of its offline counterpart for the SmartFABER dataset, and only slightly worse for CASAS. Indeed, the recognition results are similar except for *fill medication dispenser* ($ac_1$), *water plants* ($ac_3$) and *choose outfit* ($ac_8$). In case of *fill medication dispenser* ($ac_1$) and *water plants* ($ac_3$), these activities are essentially recognized by specific events that have to be temporally close. For instance, *water plants* is characterized by the events *"opening the kitchen cupboard"* and *"taking water"*. Unfortunately, our segmentation technique often separates those events in different segments as they are not exclusively related to a single activity and subjects usually performed other interleaved activities. Regarding *choose outfit* ($ac_8$), looking closely at the data, we noticed that usually this activity has a long duration and most related sensor events are also related to other activities. These facts trigger ASP3 (consistency likelihood) to initiate unnecessary segments, negatively impacting recognition rates. On the other side, the activities *watch DVD* ($ac_2$) and *prepare birthday card* ($ac_5$) are significantly better recognized by the online algorithm. Indeed, those activities can be better recognized when isolated in specific segments and separated from possibly noisy sensor events belonging to other activities. Considering the overall results on this dataset, we claim that the decrease of accuracy (at most $-2\%$) introduced by online segmentation for the CASAS dataset is sufficiently small to preserve the utility of predictions for most applications.

In order to support our choice of MLN as a classifier, we explored alternative machine learning methods, excluding purely data driven ones, since our goal is unsupervised activity recognition. We chose Hidden Markov Models (HMM) since: (i) it has been largely used in the literature for recognizing activities based on sensor data, and (ii) being based on probabilities, it can be seamlessly integrated into our reasoning architecture. Since we target unsupervised reasoning, we extract HMM parameters through semantic reasoning, using the same methods used for building our MLN knowledge base. We refer to this method as *Semantic-HMM*. In *Semantic-HMM*, the observable states are sensor events, while the hidden states are activities. The emission probabilities (i.e., probabilities of observing a sensor event given the performed activity) are our semantic correlations. The transition probabilities (i.e., probability of switching from an activity to another) are set applying the following intuition: it is more likely to continue to perform the same activity, while changing activity is less likely. Finally, the initial probabilities are equally distributed. In order to have a fair comparison, we applied the segmentation strategy that we propose in this paper, performing HMM (precisely, the Viterbi algorithm) on each segment. Even though Semantic-HMM achieves good recognition rates, our Online POLARIS outperforms it considering the overall F1-score. Indeed, our MLN model is capable of capturing complex semantic relationships and temporal aspects, while HMM only captures simple relationships. This can be observed, for instance, for the activities *answer phone* ($ac_4$) and *cooking* ($ac_{10}$), where the recognition rate of Semantic-HMM is significantly lower with respect to Online POLARIS.

Finally, in order to evaluate alternative approaches to extract semantic information about activities with respect to ontology reasoning, we implemented an alternative version of *Online POLARIS* (called *Web POLARIS*) which extracts the activity model from the Web according to a recently proposed method [22]. That method exploits computer vision tools to extract semantic information from pictures taken during the execution of a set of activities. By mining the extracted information, the technique computes the probability distribution of objects (and corresponding sensor events) over the activities. We thus replaced the semantic correlations computed by our ontology with probabilities extracted with this technique, in order to compare the two methods. The comparison with Web POLARIS indicates that the probabilities extracted from the ontology yield significantly higher recognition rates for almost every considered activity, since the ontology encodes some important relationships between home infrastructure and activities that are not captured by pictures mined from the web.

## 9 Strong points and limitations of POLARIS

POLARIS requires a relevant knowledge engineering effort to define a comprehensive ontology of activities, home environment, and sensor events. For instance, our ontology includes 235 classes and 59 properties. However, the knowledge engineering effort can be reduced by reusing existing ontologies. In particular, the ontology used in this work is an extension of the COSAR ontology [26], 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 before, and a few additional properties used by our

reasoning method. Modeling the extension required one day of work by a researcher with good skills in OWL 2. Moreover, we were able to use the same ontology for both apartments involved in our experimentation, which had very different characteristics.

However, it is questionable whether in larger scale implementations the same ontology can be adequate to cover every possible home environment and individuals' mode of activity execution. The main precondition to reuse an ontology in different environments is that it has to cover all the concepts useful for activity recognition (i.e., objects, sensor events, activities, other context data). Those concepts need to be mapped to considered activities and environmental components by means of ontological instances.

In terms of adaptability to changes in the environment and considered ADLs, our approach presents several advantages with respect to data driven ones. Indeed, differently from supervised learning methods, POLARIS can seamlessly adapt to both changes in the environment and introduction of new activities, with minimal knowledge engineering effort. In particular, novel devices can be seamlessly introduced in the ontology by adding new axioms or slightly modifying existing ones. For instance, considering Example 3, suppose that a new rice cooker is introduced in the smart home. In this case, it is sufficient to: (i) add a new instance of the ontological class RiceCooker (a subclass of CookingInstrument), (ii) modify the axiom (1 to 10) to specify that one rice cooker is in the home, and (iii) instantiate the rice cooker power sensor:

$$\{\text{s\_riceCooker}\} \equiv \text{PowerSensor} \sqcap \exists \text{sensesUsageOf}.\{\text{RiceCooker}\} \sqcap (\exists \text{producesEvent}.\{\text{et\_riceCooker}\}$$

Similarly, when a device is removed from the smart home, the above mentioned axioms must be simply modified or removed accordingly. On the contrary, most supervised learning methods cannot deal with the introduction of new data that did not appear in the original training set. Hence, those methods cannot take advantage of data provided by new devices that were not used when the training data was acquired. Besides, the introduction of new sensors and devices may enable the recognition of additional activities. For instance, a rice cooker sensor enables the recognition of a new activity "Preparing rice", that may not appear in the original ontology. In order to recognize this new activity, the ontology must be extended by introducing a new axiom:

$$\text{PrepRice} \sqsubseteq \text{PrepHotMeal} \sqcap \exists \text{requiresUsageOfArtifact}.\text{RiceCooker}$$

Note that the introduction of a new activity does not affect the ontological definition of existing ones. On the contrary, it can take advantage of existing ontological definitions. For instance, in the above example, "Preparing rice" is defined reusing the definition of "Preparing hot meal". Hence, with our approach, we reduce the burden of knowledge engineering, and we gain a clear advantage with respect to data-driven methods. Indeed, in supervised learning methods, the introduction of a new activity requires the acquisition of a completely new training set for it, which is time-consuming, costly, and unpractical.

In general, using a semantic abstraction layer by means of ontological classes (e.g., sensors, objects, home appliances, activities) strongly reduces the burden of setting up the system in a new environment with respect to supervised solutions. In order to substantiate our claims, Figure 5 shows how the recognition rate varies by increasing the number of subjects considered in the training set. Results indicate that a considerable number of subjects is required to reach results similar to the ones we obtain with our knowledge-based approach. Moreover, the required time for annotation is highly variable: it depends on the modality, on the detail of annotation, on the complexity of the activities, and on the environment. In the literature, reported annotation time varies from 30 minutes to 10 hours for each hour of activity [5], [6].

Changes in the user's health status or habits may determine a relevant concept drift that would require human intervention on the ontology. In this work we do not address this issue; however, we point out that these interventions are usually localized and limited to some abstraction levels. It is also possible to adopt a knowledge-based active learning algorithm to continuously refine a possibly generic and incomplete ontology, exploiting inhabitants feedback [37].

Finally, we want to point out that our method cannot be seamlessly re-used for any activity recognition task. For instance, consider a setup where few (or none) environmental sensors are deployed in the environment and the inhabitant is monitored through wearable devices (e.g., smartphone, smartwatch). The inertial sensors of those devices continuously stream raw sensor data about the inhabitant's movement patterns. Those data can not be directly used by our ontological framework, since it is not feasible to translate them into high-level concepts. Indeed, only a purely data-driven approach can be used to transform inertial sensor readings into higher-level information (e.g., inhabitant's posture or physical activity). However, high-level information generated by those data-driven solutions, coupled with environmental sensor data, could be possibly used by a hybrid statistical/ontological framework to better identify the complex ADLs that the user is carrying out like the ones considered in this work.

## 10 Conclusion

While most activity recognition systems adopt supervised reasoning, in this work we proposed an alternative approach, presenting an unsupervised framework for online recognition of complex ADLs. Thanks to a combination of ontological and probabilistic reasoning, our method addresses a major problem of supervised ones, namely the need to re-acquire a training set when moving the system to a new sensorised environment. A thorough experimental evalution showed that the accuracy of our unsupervised approach is comparable to the one of state-of-the-art supervised methods. In future work, we plan to investigate techniques to reduce the burden of ontological activity modeling, possibly based on activity mining from external sources, and active learning methods to fine-tune the activity model to the user's context. We also plan to extend POLARIS to the multi-inhabitant setting, by exploiting an additional module that assigns each sensor event to the subject that triggered it.

## References

[1] D. Riboni, C. Bettini, G. Civitarese, Z. H. Janjua, and V. Bulgari, "From lab to life: Fine-grained behavior monitoring in the elderly's home," in *Proc. of PerCom Workshops*. IEEE Comp. Soc., 2015, pp. 342–347.

[2] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys*, vol. 46, no. 3, pp. 33:1–33:33, 2014.

[3] T. Gu, L. Wang, Z. Wu, X. Tao, and J. Lu, "A pattern mining approach to sensor-based human activity recognition," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1359–1372, 2011.

[4] D. J. Cook, K. D. Feuz, and N. C. Krishnan, "Transfer learning for activity recognition: A survey," *Knowledge and Information Systems*, vol. 36, no. 3, pp. 537–556, 2013.

[5] K. D. Feuz and D. J. Cook, "Real-time annotation tool (RAT)," in *Proc. of AAAI Workshops*. AAAI, 2013.

[6] D. Roggen, A. Calatroni *et al.*, "Collecting complex activity datasets in highly rich networked sensor environments," in *Proc. of Intl. Conf. on Networked Sensing Systems*. IEEE Comp. Soc., 2010, pp. 233–240.

[7] 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.

[8] N. D. Rodríguez, M. P. Cuéllar, J. Lilius, and M. D. Calvo-Flores, "A survey on ontologies for human behavior recognition," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 43, 2014.

[9] G. Acampora, D. J. Cook, P. Rashidi, and A. V. Vasilakos, "A survey on ambient intelligence in healthcare," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2470–2494, 2013.

[10] J. Ye, S. Dobson, and S. McKeever, "Situation identification techniques in pervasive computing: A review," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 36–66, 2012.

[11] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, and U. Sattler, "OWL 2: The next step for OWL," *Journal of Web Semantics*, vol. 6, no. 4, pp. 309–322, 2008.

[12] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1, pp. 107–136, 2006.

[13] D. Riboni, T. Sztyler, G. Civitarese, and H. Stuckenschmidt, "Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning," in *Proc. of UbiComp*. ACM, 2016, pp. 1–12.

[14] D. Weinland, R. Ronfard, and E. Boyer, "A survey of vision-based methods for action representation, segmentation and recognition," *Computer Vision and Image Understanding*, vol. 115, no. 2, pp. 224–241, 2011.

[15] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Proc. of PERVASIVE*. Springer, 2004, pp. 1–17.

[16] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, "A hybrid discriminative/generative approach for modeling human activities," in *Proc. of IJCAI*. Morgan Kaufmann, 2005, pp. 766–772.

[17] D. Riboni, C. Bettini, G. Civitarese, Z. H. Janjua, and R. Helaoui, "SmartFABER: Recognizing fine-grained abnormal behaviors for early detection of mild cognitive impairment," *Artificial Intelligence in Medicine*, vol. 67, pp. 57–74, 2016.

[18] J. J.-C. Ying, B.-H. Lin, V. S. Tseng, and S.-Y. Hsieh, "Transfer learning on high variety domains for activity recognition," in *Proceedings of the ASE BigData & SocialInformatics 2015*. ACM, 2015, pp. 37:1–37:6.

[19] R. S. Sutton, A. G. Barto *et al.*, *Reinforcement learning: An introduction*. MIT press, 1998.

[20] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Trans Pattern Anal Mach Intell*, vol. 28, no. 4, pp. 594–611, 2006.

[21] P. Palmes, H. K. Pung, T. Gu, W. Xue, and S. Chen, "Object relevance weight pattern mining for activity recognition and segmentation," *Pervasive and Mobile Computing*, vol. 6, no. 1, pp. 43–57, 2010.

[22] D. Riboni and M. Murtas, "Web mining & computer vision: New partners for object-based activity recognition," in *Proc. of WETICE*. IEEE Comp. Soc., 2017, pp. 158–163.

[23] D. Riboni and C. Bettini, "OWL 2 modeling and reasoning with complex human activities," *Pervasive and Mobile Computing*, vol. 7, no. 3, pp. 379–395, 2011.

[24] G. Meditskos, E. Kontopoulos, and I. Kompatsiaris, "Knowledge-driven activity recognition and segmentation using context connections," in *Proc. of ISWC*. Springer, 2014, pp. 260–275.

[25] G. Okeyo, L. Chen, H. Wang, and R. Sterritt, "Dynamic sensor data segmentation for real-time knowledge-driven activity recognition," *Pervasive and Mobile Computing*, vol. 10, Part B, pp. 155–172, 2014.

[26] 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.

[27] R. Helaoui, D. Riboni, and H. Stuckenschmidt, "A probabilistic ontological framework for the recognition of multilevel human activities," in *Proc. of UbiComp*. ACM, 2013, pp. 345–354.

[28] J. Ye, G. Stevenson, and S. Dobson, "USMART: An unsupervised semantic mining activity recognition technique," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 4, no. 4, pp. 16:1–16:27, 2014.

[29] J. Wan, M. J. O'Grady, and G. M. P. O'Hare, "Dynamic sensor event segmentation for real-time activity recognition in a smart home context," *Personal and Ubiquitous Computing*, vol. 19, no. 2, pp. 287–301, 2015.

[30] S. Aminikhanghahi and D. J. Cook, "Using change point detection to automate daily activity segmentation," in *Proc. of PerCom Workshops*. IEEE Comp. Soc. , 2017, pp. 262–267.

[31] D. Triboan, L. Chen, F. Chen, and Z. Wang, "Semantic segmentation of real-time sensor data stream for complex activity recognition," *Personal and Ubiquitous Computing*, vol. 21, no. 3, pp. 411–425, 2017.

[32] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd ed. Cambridge University Press, 2010.

[33] M. Chekol, J. Huber, C. Meilicke, and H. Stuckenschmidt, "Markov logic networks with numerical constraints," in *Proc. of ECAI2016*. IOS Press, 2016, pp. 1–9.

[34] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Tracking activities in complex settings using smart environment technologies," *Int J Biosci Psychiatr Technol*, vol. 1, no. 1, pp. 25–35, 2009.

[35] O. Banos, J.-M. Galvez, M. Damas, H. Pomares, and I. Rojas, "Window size impact in human activity recognition," *Sensors*, vol. 14, no. 4, pp. 6474–6499, 2014.

[36] N. C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Pervasive and mobile computing*, vol. 10, pp. 138–154, 2014.

[37] G. Civitarese, D. Riboni, C. Bettini, Z. H. Janjua, and R. Helaoui, "Nectar: Knowledge-based collaborative active learning for activity recognition," in *Proc. of PerCom*. IEEE Comp. Soc., 2018.


**Gabriele Civitarese** is a researcher at the Computer Science department of the University of Milan. He is a member of the EveryWare laboratory since 2014. His research interests are mainly focused in the area of pervasive computing for health-care applications. In particular, he worked on human activity recognition and abnormal behaviors detection in smart-home environments. He published his results at major international conferences and journals.

**Timo Sztyler** is a researcher in computer science at the University of Mannheim and is member of the Artificial Intelligence Research Group. His main research interests lie in the field of Activity Recognition with respect to Sensor Networks and Machine Learning. In recent publications, his focus was online personalization and adaption of probabilistic recognition models. His works are published in major international pervasive computing conferences and journals.

**Daniele Riboni** is associate professor of Computer Science at the University of Cagliari. His research interests are mainly focused on context-awareness, activity recognition, knowledge management and privacy issues in pervasive and mobile computing. He served as TPC chair and TPC vice chair for different conferences and workshops in the field, including IEEE PerCom and the International Conference on Intelligent Environments (IE). His contributions appear in major conferences and journals.

**Claudio Bettini** is full professor of Computer Science at the University of Milan, where he leads the EveryWare laboratory. His research interests cover the areas of mobile and pervasive computing, data privacy, spatio-temporal data management, and knowledge reasoning. He acted as associate editor for IEEE TKDE, the VLDB Journal, and Pervasive and Mobile Computing journal. He is a IEEE senior member.

**Heiner Stuckenschmidt** is full professor of Computer Science and chairholder for Artificial Intelligence at the University of Mannheim. His group performs fundamental and applied research in knowledge representation and reasoning with a focus on combining logical and probabilistic reasoning. His recent interests include the application of supervised and unsupervised methods to the problem of human activity recognition.