Pervasive and Mobile Computing 56 (2019) 88–105

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

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

newNECTAR: Collaborative active learning for knowledge-based probabilistic activity recognition[☆]

Gabriele Civitarese[a,*], Claudio Bettini[a], Timo Sztyler[b], Daniele Riboni[c], Heiner Stuckenschmidt[b]

[a] University of Milano, Milano, Italy
[b] University of Mannheim, Mannheim, Germany
[c] University of Cagliari, Cagliari, Italy

A R T I C L E I N F O

Article history:
Available online 16 April 2019

Keywords:
Pervasive computing
Activity recognition
Active learning
Temporal pattern mining

A B S T R A C T

The increasing popularity of ambient assisted living solutions is claiming adaptive and scalable tools to monitor activities of daily living. Currently, most sensor-based activity recognition techniques rely on supervised learning algorithms. However, the acquisition of comprehensive training sets of activities in smart homes is expensive and violates the individual's privacy. In this work, we address this problem by proposing a novel hybrid approach that couples collaborative active learning with probabilistic and knowledge-based reasoning. The rationale of our approach is that a generic, and possibly incomplete, knowledge-based model of activities can be refined to target specific individuals and environments by collaboratively acquiring feedback from inhabitants. Specifically, we propose a collaborative active learning method exploiting users' feedback to (i) refine correlations among sensor events and activity types that are initially extracted from a high-level ontology, and (ii) mine temporal patterns of sensor events that are frequently generated by the execution of specific activities. A Markov Logic Network is used to recognize activities with probabilistic rules that capture both the ontological knowledge and the information obtained by active learning. We experimented our solution with a real-world dataset of activities carried out by several individuals in an interleaved fashion. Experimental results show that our collaborative and personalized active learning solution significantly improves recognition rates, while triggering a small number of feedback requests. Moreover, the overall recognition rates compare favorably with existing supervised and unsupervised activity recognition methods.

© 2019 Elsevier B.V. All rights reserved.

1. Introduction

Sensor-based human activity recognition has been a hot topic in pervasive computing for several years for its important applications in assisted living, e-health, and context-aware services. While simple activities like running or cycling can be easily recognized even only by acceleration data, the research has focused on the recognition of complex activities of daily living (ADL) [1,2].

☆ A preliminary version of this work appeared in the Proceedings of the IEEE International Conference on Pervasive Computing (PerCom 2018).
* Corresponding author.
E-mail addresses: gabriele.civitarese@unimi.it (G. Civitarese), claudio.bettini@unimi.it (C. Bettini), timo@informatik.uni-mannheim.de (T. Sztyler), riboni@unica.it (D. Riboni), heiner@informatik.uni-mannheim.de (H. Stuckenschmidt).

https://doi.org/10.1016/j.pmcj.2019.04.006
1574-1192/© 2019 Elsevier B.V. All rights reserved.

Most ADL recognition systems rely on supervised learning which requires a large training set of sensor events with annotations mapping sequences of events to the specific activity that generated them. Annotation can be done by direct observation, video recording, crowdsourcing, or by manual data analysis, but in all cases it is an expensive process. Besides privacy issues in observing individuals in their private spaces, the heterogeneity of the environments and the very diverse ways a complex activity can be performed by different individuals limit the value of the acquired training set. This is a serious limitation to the large scale deployment of these systems. Active learning has been proposed to mitigate this problem [3], but the majority of these techniques need anyway a starting labeled training set. Alternative approaches propose the use of a structured knowledge representation of activities, infrastructure, and events to guide the recognition process in an unsupervised way [4]. In order to be effective, they require a significant effort of knowledge engineers to build a comprehensive ontology, and it remains questionable if such an ontology could actually cover an heterogeneous large set of environments and individuals.

We address this problem by a novel framework, called *newNECTAR*, exploiting *kNowledge-basEd Collaborative acTive learning for Activity Recognition*. *newNECTAR* does not require an initial training set, since it relies on a (possibly incomplete) ontology to derive a first set of semantic correlation values between activities and the sensor events that most likely will be observed when performing those activities. These correlations together with a small set of general inference rules are formally expressed as probabilistic formulas in a Markov Logic Network (MLN). Given a sequence of observed sensor events, the MLN inference mechanism identifies the most probable sequence of ADLs that generated them. In order to cope with the incompleteness of the ontology and the heterogeneity of environments and individuals, we introduce a collaborative active learning process to refine the correlations derived by the ontology and the probabilistic rules of the MLN. The stream of sensor events is segmented in real-time, and based on the uncertainty about the activity associated with the events in the segment, a feedback may be asked to the subject about which activity is being performed. Feedback responses coming from different homes are collected and analyzed in a cloud infrastructure; in particular, the system identifies frequent associations between activities and specific events or temporal patterns of events. Based on this analysis, each home receives personalized information to refine or insert new probabilistic rules in its MLN, hence refining the recognition model. The collaborative active learning feature of *newNECTAR* also deals with the common situation in which a new device is installed in the infrastructure, by producing a new set of correlation values regarding the new device events.

newNECTAR has been experimentally validated on a real world and publicly available dataset involving multiple subjects performing ADLs in both sequential and interleaved fashion.

The main contributions can be summarized as follows:

- We propose a new active learning approach to ADL recognition that addresses the main problems of current statistical and knowledge-based methods;
- The *newNECTAR* technique supports collaborative and personalized refinement of the recognition model based on the temporal data analysis of the user feedbacks;
- Our experiments show the gain obtained by collaborative active learning, the moderate effort required to the involved subjects, and the overall effectiveness of *newNECTAR* even compared with supervised approaches.

The rest of the paper is structured as follows. Section 2 summarizes related work. Section 3 presents the *newNECTAR* architecture, while the techniques are explained in Section 4. Section 5 shows experimental results and Section 6 discusses open issues. Finally, Section 7 concludes the paper.

2. Related work

Acquiring comprehensive training sets of ADLs is expensive in terms of annotation costs and it involves non trivial issues related to the privacy of the participating subjects [5]. Hence, several efforts have been devoted to devise unsupervised or semi-supervised activity recognition techniques [6–8], as well as transfer learning methods for activity models [9,10].

Unsupervised activity recognition methods do not require the acquisition of labeled training sets. As a consequence, the model of activities must be either manually specified (e.g., through an ontology) or mined from other sources (e.g., Web resources, or unlabeled datasets of activities). A popular approach to the manual specification of activity models consists in the use of description logics or logical rules to define the formal semantics of activities [6–8]. In those approaches, complex activities are defined in terms of their simpler components. Sequences of simple actions, recognized based on firing of specific sensor events, are matched to activity definitions to identify the occurred activity. However, those approaches rely on rigid assumptions about the execution patterns of ADLs [11]. On the contrary, complex activities are characterized by large variability of execution. In order to cope with that issue, other works investigated the use of less rigid formalisms to define ADLs. In [12], probabilistic description logics are used to define a multi-level ontology of domestic activities. Hybrid ontological-probabilistic reasoning is used in [4] to recognize ADLs based on semantic correlations among sensor events and activities. However, all those approaches require significant knowledge engineering efforts, and are hardly scalable to the definition of a comprehensive set of ADLs in different contexts.

A different approach consists in mining activity models from Web resources [13–15]. Those methods analyze textual descriptions of activities mined from the Web in order to obtain correlations among activities and objects used for their execution. That approach has been recently extended to exploit visual cues extracted from the Web, such as images and videos [16]. However, it is questionable whether object–activity correlations are sufficient to recognize complex ADLs. Indeed, in this work we rely not only on correlations, but also on temporal constraints on event occurrences, and on knowledge-based constraints on activity components.

A further approach is to infer activity models from unlabeled datasets. For example, in [17] multi signal motifs mining on data acquired from body-worn sensors is used to recognize repeated subsequences of sensor data, that represent the execution of a specific (unknown) activity. A similar approach, but applied to domestic sensors, is proposed in [18]. In [19], data mining methods are used to cluster sequences of sensor events, such that each cluster represents an activity class. The inhabitant is asked to provide the actual class of each cluster. In USMART [20], a knowledge-based method is used to segment sensor event traces. Those segments are fed to a sequential pattern mining algorithm to derive frequent sequences, which are thus associated with one or more activities exploiting semantic based reasoning and then used for activity recognition. Compared to those unsupervised methods, in our work we exploit collaborative users' feedback to assign a certain semantics to sets of sensor events.

Semi-supervised learning methods use unlabeled data to improve the model computed through a training set. Different semi-supervised methods, including the ones presented in [21–23], address the recognition of physical activities based on accelerometer data. Self-training and co-training are used in [24] for recognition of ADLs. The same work also investigates the use of active learning, with the objective of asking the individuals to label the *most informative* sequences of sensor events. In [25], the authors propose to use active learning to dynamically adapt the recognition model to the changes of the home environment. In that work, an entropy based measurement is used to query the most informative sequences of sensor events to update a Dynamic Bayesian Network. An active learning method to iteratively refine the annotations of video provided by crowdsourcing services (like Mechanical Turk) is presented in [26]. That method relies on confidence scores about the annotation. A similar approach is proposed in [27]. The three methods presented in [26] to choose the most informative data points are based on (i) low confidence for the most probable activity class, (ii) small difference between the confidence of the most and second most probable class, or (iii) high entropy among the probability of classes. Experimental results in smart home settings show that the three methods achieve similar accuracy. The work presented in [3] proposes strategies to select the most appropriate annotators in a crowdsourcing framework for active learning of ADLs. Differently from those works, in this paper we propose *collaborative* active learning to share the burden of providing ADLs labels among a community of inhabitants.

Other works propose transfer learning methods to reuse activity datasets acquired in different environments [9,28]. However, effective portability of activity datasets is challenging, since datasets of complex ADLs are strongly coupled to the environment in which they are acquired and to the mode of execution of the individual [10]. Hence, in our work we consider a similarity measure between the context of the target environment (characteristics of home and inhabitant) and the one of the environment from which the label is acquired, in order to cope with the different context conditions. A related issue is how to dynamically adapt the recognition system to changes in the sensor infrastructure. With this regard, a technique was proposed to update the model of a supervised machine learning algorithm with features of newly-discovered sensors [29]. In our work, we integrate information about new sensors thanks to a collaborative active learning method.

Finally, we mention that extracting temporal knowledge from time series is a well-known and explored research area [30]. Association rules mining has been successfully applied to sensor networks data in order to derive relevant event patterns [31]. Emerging patterns discovery [32] was proposed to recognize sequential, interleaved and concurrent activities. In the field of activity discovery, spatio-temporal frequent pattern mining [33] and stream mining [34] techniques have been explored. In our work, we exploit *collaborative* active learning to mine from inhabitants' feedback the most relevant associations between quantitative temporal constraints of sensor events and activities in order to improve a possibly incomplete probabilistic knowledge-based activity model. Moreover, we personalize the mining approach by considering the characteristics of individuals and environment.

With respect to our previous work [35], we significantly improved our collaborative active learning framework by mining personalized temporal patterns of sensor events from the feedback.

3. newNECTAR's model and architecture

In this section, we present the data model and the system architecture of *newNECTAR*.

3.1. Model

In the following, we explain how we model sensor and activity data in *newNECTAR*, including temporal aspects and semantic correlations.

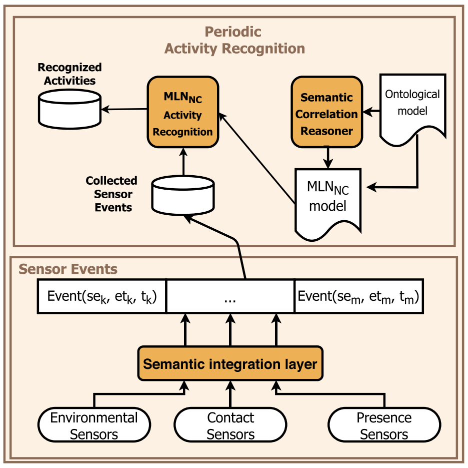


Fig. 1. Periodic activity recognition.

3.1.1. Activities and sensor events

In our model, we distinguish between an **activity class** and an **activity instance**, where the former is an abstract activity (e.g., taking medicine) and the latter is the actual occurrence of an activity of a given class during a certain time period. We denote with $\mathbf{A}$ the set of the considered activity classes (e.g., $\mathbf{A} = \{Eating, Cooking, Taking\ Medicines\}$).

We denote with $\mathbf{E}$ the set of high-level **event types** that correspond to the set of monitored operations (e.g., $\mathbf{E} = \{\ opening\_the\_medicine\_drawer,\ closing\_the\_medicine\_drawer\ \}$). In addition, $\mathbf{T}$ describes the set of all possible **event timestamps**. The set $\mathbf{T}$ is isomorphic to the set of natural numbers $\mathbb{N}$. We compute the temporal distance between two sensor events se_i and se_j as their timestamps' difference $t_i - t_j$; The temporal distance is an integer representing the number of seconds between the occurrence of se_j and the one of se_i. Note that the distance will be a negative value if se_j occurred after se_i.

A **temporal sequence of events** is represented as:

$$\langle\, se_1 = \langle et_1, t_1\rangle, \ldots, se_k = \langle et_k, t_k\rangle\,\rangle,$$

where $se_i = \langle et_i, t_i\rangle$ indicates that se_i is an instance of the event type $et_i \in \mathbf{E}$ occurred at timestamp $t_i \in \mathbf{T}$.

3.1.2. Binary event temporal patterns

Given two variables X and Y with values in event timestamps, and two integers min and max, a quantitative binary **temporal constraint** C is represented as $C = (min \le X - Y \le max)$. A temporal constraint C overlaps a temporal constraint C' defined in terms of min' and max' if the intersection of the intervals $[min, max]$ and $[min', max']$ is non-empty. We denote as $t \in C$ an event timestamp t that occurs within a temporal constraint C. A pair of sensor events (se_i, se_j) satisfies C if the inequality holds when substituting X and Y with t_i and t_j, respectively. A **binary event temporal pattern** is a tuple $\langle et_i, et_j, C\rangle$, where $et_i, et_j \in \mathbf{E}$ and C is a temporal constraint. We denote by **ETP** the set of possible event temporal patterns. A temporal sequence S of events is said to *match the pattern* $ETP = \langle et_i, et_j, C\rangle$ if S contains a pair of sensor events of types et_i and et_j whose respective timestamps satisfy C. A binary event temporal pattern $\langle et_i, et_j, C\rangle$ overlaps a binary event temporal pattern $\langle et_k, et_l, C'\rangle$ if C overlaps C', $et_i = et_k$ and $et_j = et_l$.

3.1.3. Semantic correlations

Semantic correlations represent probabilistic dependencies among event types and activity classes. Formally, given the type of a sensor event and an activity class, the **semantic correlation** function $SC : \mathbf{E} \times \mathbf{A} \rightarrow [0, 1]$ gives the probability that the event is generated by performing an instance of that activity class. Hence, given any event type, we have that SC is a probability distribution over all activity classes:

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

3.2. Hybrid activity recognition method

The architecture of our hybrid activity recognition method is shown in Fig. 1. Like in our previous work [4], we define the semantics of activities and high-level events in an OWL 2 ontology. In particular, the ontology includes axioms stating that an instance of a given activity class (e.g., "prepare soup") must necessarily generate a set of high-level events (e.g., "take water", "pour water"). Moreover, it also includes other common-sense axioms regarding time and location;

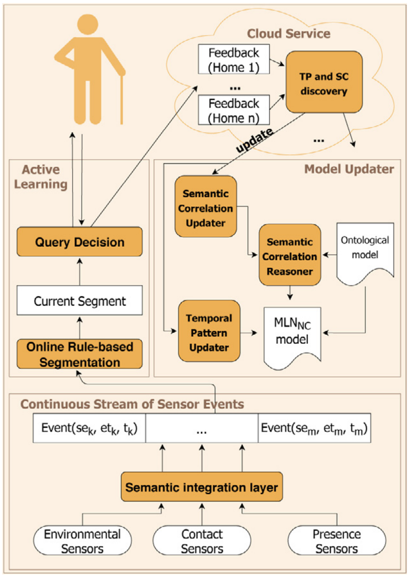


Fig. 2. The collaborative active learning mechanism of *newNECTAR*.

e.g., "every instance of cooking is executed in the kitchen". Ontological reasoning is used offline to derive *semantic correlations* based on the smart home setup. Periodically (e.g., daily), the *MLN_{NC}* ACTIVITY RECOGNITION layer is in charge of recognizing performed activities by modeling and reasoning with detected events and semantic correlations through an extension of Markov Logic Networks with numerical constraints [36].

3.3. Collaborative active learning architecture

The architecture of our collaborative active learning system is shown in Fig. 2. We assume a set of different smart-homes equipped with unobtrusive sensing infrastructures. Environmental sensors are deployed in each home in order to monitor the interaction of the inhabitant with home artifacts but also context conditions (e.g., temperature) and presence in certain locations. Each home may have a different set of deployed sensors and monitored items. A gateway in the home is in charge of collecting and pre-processing raw data from the sensor network in order to reconstruct the most probable activities that generated them.

3.3.1. Semantic integration

The SEMANTIC INTEGRATION LAYER is in charge of applying pre-processing rules in order to detect high-level events from raw sensor signals. For example, if at time *t* the medicine drawer sensor produces the raw event *open*, then the high-level event at *t* is *opening the medicine drawer*. Reasoning on high-level events allows our system to be robust to minor changes in activity execution. For instance, if the inhabitant decides to keep his/her medicines in a different drawer than before, it is sufficient to adjust the above mentioned pre-processing rules, thus adapting the mapping between raw sensors data and events.

3.3.2. Active learning

Being manually designed by knowledge engineers with a specific application in mind, the ontological model is necessarily limited to specific environments and activities. Thus, semantic correlations may not be sufficiently comprehensive to cover different application domains. Moreover, some sensor event types (e.g., motion sensors) do not convey any explicit semantic information; hence, no semantic correlation can be inferred for these event types from the ontology. Since OWL 2 does not natively support temporal features, our ontology also lacks information about event temporal patterns that are frequently associated to the execution of given activities. For these reasons, our system collects *feedback items* from the smart-homes in order to: (i) discover temporal event temporal patterns as well as semantic correlations not inferred from the ontology, and (ii) refine the values of existing correlations. For acquiring a feedback, *newNECTAR* interactively queries the user to provide the class of his/her current activity. Acquired feedback is collaboratively shared among the smart-homes to update semantic correlation values in a personalized fashion. For the sake of clarity, in the following we name *origin* the environment (home and inhabitant) providing feedback, and *target* the environment where feedback is used.

3.3.3. Activity segmentation

The feedback acquisition mechanism relies on the concept of *segments*. Formally, a segment $\vec{s} = \langle se_j, \ldots, se_k \rangle$ is a temporal sequence of *consecutive* sensor events. Each event is assigned to exactly one segment. In order to cope with interleaved activities, a single activity instance can span multiple segments. Each segment belongs to exactly one activity instance. The class of each segment is the one of its activity instance. As explained below, when *newNECTAR* determines that a segment's events do not provide enough hints to reliably determine its class according to an information-theoretic metric, it queries the user to obtain a feedback.

To this purpose, the ONLINE RULE-BASED SEGMENTATION layer is in charge of segmenting the continuous stream of sensor events. The segmentation method is based on semantic rules that consider different aspects like time constraints, objects interaction, and change of location. The role of these rules is to group together those consecutive events which most likely originate from the same activity instance. As soon as a segment is finalized, it is processed by the QUERY DECISION layer in order to decide whether triggering a feedback query or not. That module processes the segment to apply an information-theoretic metric considering the segment's events and the semantic correlations. If the activity class is uncertain according to that metric, the module triggers a feedback query. A user-friendly and unobtrusive interface is in charge of issuing the feedback query and collecting the inhabitant answer.

3.3.4. TP And SC discovery

The acquired feedback is transmitted to a CLOUD SERVICE, where the TP AND SC DISCOVERY layer is in charge of: (a) mining event temporal patterns (TP) that are frequently correlated to the execution of certain activities, and (b) discovering reliable and personalized semantic correlations (SC) between event types and activities. Personalization is based on the similarity between the origin and target environment. The cloud service periodically sends personalized feedback items to each target. Received feedback is used by the SEMANTIC CORRELATIONS UPDATER layer to discover novel semantic correlations and to update the values of existing ones. Moreover, the TEMPORAL PATTERN UPDATER is in charge of updating the MLN model to take advantage of discovered temporal event patterns.

For the sake of this work, we assume that the CLOUD SERVICE is trusted. However, in a real deployment it would likely be a *honest-but-curious* third party. Proper privacy techniques are thus needed to protect sensitive data and at the same time to preserve the CLOUD SERVICE functionalities.

4. newNECTAR under the hood

In this section, we describe in detail each component of our system.

4.1. Activity recognition

Our activity recognition system relies on an ontological model of home environment, sensors, activities and actors, and on probabilistic reasoning through an extension of Markov Logic Networks (MLN) [37].

4.1.1. Ontological model

Our ontology has been defined using the OWL 2 language. For the sake of space, we omit technical details about the ontological model and ontological reasoning methods, which can be found in [4]. In the following, we outline the main characteristics of our ontology. Ontological properties describe relations among instances. For example, an ontological axiom states that "firing of an accelerometer *a* attached to a kitchen chair *c* indicates the occurrence of an action of class *MoveKitchenChair*". In our ontology, we express necessary conditions for a set of sensor events to be generated by a given activity. For example, the sensor events generated by an instance of activity class *PrepareHotMeal* must include an event of class *UsingCookingInstrument*. Other ontological axioms describe temporal and location-based constraints. As explained below, by reasoning with those ontological axioms, we can infer generic semantic correlations among sensor event types and activity classes, which are used by the MLN module to associate sensor events to their activity instance.

4.1.2. Semantic correlation reasoner

Inference of semantic correlations relies on the property composition operator of OWL 2 [11]. In particular, in the ontology, we defined an axiom stating that: "if an event of type *et* is produced by a sensor that indicates the usage of an artifact possibly used for an activity of class *ac*, then *et* is a **predictive sensor event type** for *ac*". For each event type *et*, we compute the set of activities for which *et* is a predictive event type:

$$predAct(et) = \{ac \mid et \text{ is a predictive event for } ac\}$$

To enforce property (1), we set the values of the semantic correlation function *SC* 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:

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

Otherwise, having no information about the associations between *et* and activity classes, we uniformly distribute its correlation values to all possible activity classes:

$$SC(et, ac) = \frac{1}{|\mathbf{A}|},$$

where **A** is the set of all activity classes. It is easy to verify that in both cases property (1) (see Section 3.1.3) is enforced.

4.1.3. MLN activity recognition

MLN is a probabilistic first-order logic that naturally supports reasoning with uncertain axioms and facts [37]. In our framework, we use an extension of MLN, named MLN_{NC}, which supports numerical constraints useful to reason with temporal information. In the following, we sketch the main inference tasks for activity recognition; more details can be found in [4]. MLN supports the definition of both **hard and soft axioms**. The former are certainly true, and are mainly automatically extracted from our ontology. The latter are associated to a weight that represents their probability of being true, considering the inferred semantic correlations. We instantiate the MLN knowledge base by translating the axioms of our ontology in hard MLN axioms. In this way, we ensure that the MLN_{NC} knowledge base is consistent with our OWL 2 ontology. Moreover, we add soft MLN axioms to represent common-sense knowledge about typical activity execution. In particular, the extension of MLN with numerical constraints allows us to express probabilistic common-sense knowledge about the typical time during which activities are executed. Similarly, we define soft axioms about the maximum and minimum duration of activities. At activity recognition time, we add facts that represent the observation of occurred sensor events. Each fact $SensorEvent(e, et, t)$ includes the event unique identifier e, the event type et and the timestamp t. We also add facts regarding initial hypothesis of activity instances, which are computed by a heuristic algorithm considering semantic correlations. Each fact $ActClass(ai, ac)$ includes the corresponding candidate activity instance identifier ai and its most likely activity class ac.

We add probabilistic axioms that relate sensor events to candidate activity instances according to the semantic correlation values of the corresponding event type and activity class. For instance, the probabilistic axiom:

$$\begin{aligned}&-0.619\ BelongsTo(\text{'}PourWater\text{'}, 1029, ai)\\&\quad\wedge ActClass(ai, \text{'}PrepareSoup\text{'})\end{aligned}$$

states that, with weight −0.619, the sensor event of type ‘PourWater’ observed at timestamp 1029 belongs to an activity instance *ai* of class ‘PrepareSoup’. According to the MLN semantics of weights, the axiom weight is computed applying the *logit* function to the corresponding probability value obtained by the semantic correlation function. For instance, the value −0.619 in the above formula is obtained by applying *logit* to 0.35, which is the value of $SC(\text{'}PourWater\text{'}, \text{'}PrepareSoup\text{'})$.

We also include probabilistic axioms to represent *temporal patterns*. In particular, given a binary event temporal pattern $\langle et_i, et_j, C\rangle$ frequently associated to the execution of activity ac and $C = (min \le X - Y \le max)$, we add a soft rule:

$$\begin{aligned}&w\ SensorEvent(e_i, et_i, t_i) \wedge SensorEvent(e_j, et_j, t_j)\ \wedge\\&\quad t_i - t_j \ge min\ \wedge\ t_i - t_j \le max \rightarrow\\&\quad OccurredIn(e_i, ai)\ \wedge\ OccurredIn(e_j, ai)\ \wedge\ ActClass(ai, a)\end{aligned}$$

where $w > 0$ is the soft rule's weight, e_i, e_j are sensor events occurred at t_i, t_j respectively, and ai is an activity instance. For example, suppose that the temporal pattern consisting in the sequence of two events of type FridgeOpened and StoveTurnedOn occurring at a temporal distance ranging from 5 to 25 s is frequently associated to the execution of Cooking. That binary event temporal pattern is encoded by the following rule:

$$\begin{aligned}&0.15\ SensorEvent(e_i, \text{'}FridgeOpened\text{'}, t_i)\ \wedge\\&\quad SensorEvent(e_j, \text{'}StoveTurnedOn\text{'}, t_j)\ \wedge\ t_i - t_j \ge 5\ \wedge\ t_i - t_j \le 25 \rightarrow\\&\quad OccurredIn(e_i, ai)\ \wedge\ OccurredIn(e_j, ai)\ \wedge\ ActClass(ai, \text{'}Cooking\text{'})\end{aligned}$$

The weight w of the rule corresponding to an *etp* is chosen using the method explained in Section 4.3.3.

Periodically (e.g., daily) our system analyzes the sensor events collected in that period to generate the MLN_{NC} model as described above. Finally, the *MLN* reasoning mechanism executes maximum a-posteriori inference to compute the most probable assignment of (i) sensor events to activity instances, and (ii) activity class to activity instances.

4.2. Online segmentation and query decision

The segmentation of the continuous stream of sensor data is performed by the Online rule-based segmentation layer based on knowledge-based conditions. For each produced segment, the query decision layer thus computes its entropy considering the semantic correlations. If the entropy of a segment exceeds a fixed threshold, the inhabitant is queried in order to provide an activity class to the segment. This allows us to limit the number of queries issued to the inhabitant.

4.2.1. Online rule-based segmentation

The ONLINE RULE-BASED SEGMENTATION layer is in charge of deciding whether it is appropriate to finalize the current segment and initiate a new one. For that purpose, it considers semantic conditions in order to interpret the continuous stream of sensor events. This includes the observation of interactions with objects (*C1*), changes between rooms (*C2*), and unusual gaps in time between consecutive sensor events (*C3*). Whenever a new sensor event e_{new} is observed, all these three conditions are reviewed. If at least one of the conditions is fulfilled the current segment is finalized. Hence, the sensor event e_{new} is the first element of the new segment. Of course, the objective is to reconstruct the ground truth segments based on the observed stream of sensor events. In the following, we describe the mentioned conditions in detail:

(C1) This condition keeps track of the events that result from interaction with the objects in the home. If at a certain point in time the subject stops to interact with all items, we consider this as an indicator that an activity instance was completed and thus the current segment is finalized.

(C2) Several activities are bound to a certain location or room. For that reason, if sensor events show that the inhabitant moves from a room to another, C2 considers this situation as an indicator to finalize the current segment.

(C3) An increasing temporal distance between consecutive sensor events can be interpreted as a reduced probability that they describe the same activity instance. This is especially the case if the subject leaves the observed home. Therefore, we keep track of the median distance between sensor events collected in the previous days, and we assume that two consecutive sensor events belong to different segments if their temporal distance is twice as large as the median. When no sufficient statistical information about previous sensor events is available, we use a manually fixed threshold.

These conditions aim to generate segments which cover at most one activity instance: we prefer to split an activity instance in more 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. Indeed, we want to reduce the risk of associating the user's answer with wrong sensor events. Moreover, this segmentation strategy allows us to cope with interleaved activities.

As soon as a new segment is generated, it is forwarded to and processed by the QUERY DECISION layer.

4.2.2. Query decision

Given a segment $\vec{s}$, the QUERY DECISION layer decides if it is necessary to query the inhabitant. In particular, if the semantic correlations regarding the events in $\vec{s}$ are inconclusive when considered together (i.e., they do not converge on a specific activity class), we ask the inhabitant which activity he/she was actually performing. For that purpose, we introduce the concept of a **segment's bag**:

$$Bag(\vec{s}) = \{et \mid se = \langle et, t\rangle \in \vec{s}\}$$

where $\vec{s}$ is a finalized segment and $Bag(\vec{s})$ is a bag (i.e., a multiset) which contains the types of the events contained in $\vec{s}$. It is important to note that the temporal order of events of a segment is not reflected by its bag. Hence, for each bag $Bag(\vec{s})$, we compute for all the activity classes $ac \in \mathbf{A}$ the **likelihood** that the segment $\vec{s}$ represents an activity instance of ac. This is computed as follows:

$$L(ac \mid \vec{s}) = \frac{\sum_{et \in Bag(\vec{s})} SC(et, ac)}{|Bag(\vec{s})|}$$

where $SC(et, ac)$ is the semantic correlation between et and ac.

After we compute $L(ac \mid \vec{s})$ for all activity classes, we normalize these values in order to have a probability distribution. Subsequently, the **segment's entropy** is calculated on the distribution to determine the system's confidence for the segment:

$$H(\vec{s}) = \sum_{ac \in A} P(X = ac \mid \vec{s}) \cdot log(\frac{1}{P(X = ac \mid \vec{s})})$$

where $P(X = ac \mid \vec{s})$ results from the normalized $L(ac \mid \vec{s})$ values.

Finally, if $H(\vec{s})$ is higher than a predefined threshold λ, the system ranks $\vec{s}$ as *uncertain*. In this case, the system queries the inhabitant in order to provide an activity label ac for $\vec{s}$. Thus, the system transmits immediately the **segment feedback item** $\langle \vec{s}, ac, o\rangle$ to the CLOUD SERVICE, where o is the identifier of the origin.

Note that segments containing noisy events that occurred outside activities execution (e.g., trigger of presence sensors) would likely lead to high entropy values. To overcome this issue, we rely on the SEMANTIC INTEGRATION LAYER presented in Section 3 to reduce as much as possible the generation of those noisy events, filtering out events that are not relevant to activity recognition according to our ontological description of activities. Moreover, we also discard segments with few events in order to further reduce noisy data; in particular, segments whose length is less than a threshold γ are discarded.

4.3. Collaborative adaptation

In the following, we describe our collaborative adaptation framework, which relies on three main components. The TP AND SC DISCOVERY layer (which runs on the CLOUD SERVICE) collects and aggregates the *feedback* received from origin homes and it periodically transmits personalized updates to each target home. On the other hand, the SEMANTIC CORRELATION UPDATER and TEMPORAL PATTERN UPDATER, which run in the home's gateway, are in charge of using personalized feedback items to refine the values of semantic correlations and temporal patterns, respectively.

4.3.1. TP and SC discovery

The TP AND SC DISCOVERY layer is in charge of computing personalized feedback items based on the received feedback. Personalization is based on the similarity between the origin and target of a feedback. In order to measure the similarity, that module relies on a *similarity function* $sim : H \times O \to [0, 1]$, where H is the set of targets, and O is the set of origin environments. Of course, the most appropriate definition of the target environment features, as well as the method to compute *sim* values, depend on the addressed application.

Personalized SC feedback items. At first, the module preprocesses each segment feedback item $\langle \vec{s}, ac, o\rangle$ received from the participating homes, to obtain a multiset of **SC feedback items**. Each SC feedback item $f = \langle et, ac, o\rangle$ corresponds to a sensor event of the vector, where et is the event type, ac and o are the activity class and the origin of the segment feedback item, respectively.

Example 1. Consider the segment feedback item $\langle \vec{s}, ac, o\rangle$, where:

$$\vec{s} = \langle \langle et_1, t_1\rangle, \langle et_2, t_2\rangle, \langle et_3, t_3\rangle\rangle.$$

That item is transformed in the following multiset of SC feedback items:

$$(\langle et_1, ac, o\rangle, \langle et_2, ac, o\rangle, \langle et_3, ac, o\rangle).$$

Based on the multiset F of SC feedback items obtained aggregating the items received from all the participating homes, the module computes personalized SC feedback items for each target environment. In particular, consider a target h. At first, for each event type et and activity class ac, the following formula computes the personalized SC feedback **support**:

$$supp(et, ac, h, F) = \sum_{f=\langle et,ac,o\rangle \in F} sim(h, o).$$

In order to exclude unreliable feedback, the CLOUD SERVICE transmits only personalized SC feedback items whose support is larger than a threshold σ. For each reliable personalized feedback, the module computes its **predictiveness** value:

$$pred(et, ac, h, F) = \frac{supp(et, ac, h, F)}{\sum_{ac_i \in \mathbf{A}} supp(et, ac_i, h, F)},$$

which is the normalization of et's support values, distributed over all the activity classes. The module also computes estimated **similarity** as the median value of the similarity between the SC feedback items' origin and the target:

$$s(et, ac, h, F) = \underset{f=\langle et,ac,o\rangle \in F}{median}\ sim(h, o).$$

Finally, the system communicates to the target home each **personalized SC feedback item**:

$$\langle et, ac, p, s\rangle,$$

where et is an event type, ac is an activity class, p is the predictiveness, and s is the similarity.

Personalized TP feedback items. The cloud service also analyzes the collected segment feedback items to obtain a set of **temporal pattern (TP) feedback items**. A **TP feedback item** is a tuple $\langle etp, ac\rangle$, where etp is a binary event temporal pattern and ac is the associated activity class. In the following, we explain how we mine segment feedback items to infer TP feedback items. Let **S** be the set of all the segment feedback items collected by the origin homes. We assume that the set **E** of the event types is totally ordered (e.g., lexicographically). An **event pair occurrence** $\langle et_i, et_j, d, o\rangle$, where $et_i \le et_j$, represents the occurrence of two distinct sensor events $ev(et_i, t_i)$, $ev(et_j, t_j)$ within the same segment provided by an origin home o with a temporal distance $d = t_i - t_j$. The multi-set of event pair occurrences given a particular activity class ac is denoted as follows:

$$epo(ac) = \bigcup_{S=\langle \vec{s},ac,o\rangle \in \mathbf{S}} \{\langle et_i, et_j, t_i - t_j, o\rangle | ev(et_i, t_i), ev(et_j, t_j) \in \vec{s}, et_i \le et_j\}$$

Given a pair of event types $et_i, et_j \in \mathbf{E}$ and an activity class ac, the **broadest event temporal pattern** $betp(et_i, et_j, ac)$ is an event temporal pattern $\langle et_i, et_j, C\rangle$ such that C is a binary quantitative temporal constraint bounded by the minimum and the maximum temporal distances found in the event pair occurrences $epo(ac)$:

$$betp(et_i, et_j, ac) = \langle et_i, et_j, \min_{\langle et_i, et_j, d, o\rangle \in epo(ac)} d \leq X - Y \leq \max_{\langle et_i, et_j, d, o\rangle \in epo(ac)} d\rangle$$

where X and Y are two variables with values in event timestamps. Hence, we compute the set $tp(ac)$ of TP feedback items for a particular activity class ac in the following way:

$$tp(ac) = \{\langle betp(et_i, et_j, ac), ac\rangle \mid \forall \langle et_i, et_j, d, o\rangle \in epo(ac)\}$$

The support of a TP feedback item $\langle etp, ac\rangle$ for a target home h can be computed as:

$$supp(\langle etp = \langle et_i, et_j, C\rangle, ac\rangle, h) = \frac{\sum_{\langle et_i, et_j, d, o\rangle \in epo(ac)} sim(h, o)}{\sum_{\langle et_l, et_m, d_n, o'\rangle \in \bigcup_{ac_k \in \mathbf{A}} epo(ac_k)} sim(h, o')}$$

Intuitively, the support of a TP feedback item $\langle etp = \langle et_i, et_j, C\rangle, ac\rangle$ is the percentage of event pair occurrences where et_i, et_j occurred together in the same instance of ac, scaled by the similarity between target and origins' homes. To exclude unreliable feedback, the Cloud Service transmits only TP feedback items whose support is larger than a threshold σ'.

The Cloud Service further refines the TP feedback items whose support is larger than σ'. To this aim, it groups together all those items whose binary event temporal patterns overlap. From these items, it generates **refined TP feedback items**, whose binary constraints form a partition of the binary constraints of the original TP feedback items.

Example 2. Consider two TP feedback items $\langle\langle OpenFridge, CloseFridge, 1 \leq X - Y \leq 6\rangle, Cooking\rangle$ and $\langle\langle OpenFridge, CloseFridge, 4 \leq X - Y \leq 20\rangle, Cleaning\rangle$. It is easy to see that the event patterns of the two feedback items overlap. These items are replaced with refined TP feedback items that essentially partition the overlapping binary constraints of the original items:

- $\langle\langle OpenFridge, CloseFridge, 1 \leq X - Y \leq 3\rangle, Cleaning\rangle$
- $\langle\langle OpenFridge, CloseFridge, 4 \leq X - Y \leq 6\rangle, Cooking\rangle$
- $\langle\langle OpenFridge, CloseFridge, 4 \leq X - Y \leq 6\rangle, Cleaning\rangle$
- $\langle\langle OpenFridge, CloseFridge, 7 \leq X - Y \leq 20\rangle, Cooking\rangle$

For each refined TP feedback item, the system computes the **confidence**:

$$conf(\langle\langle et_i, et_j, C\rangle, ac\rangle) = \frac{|\{\langle et_i, et_j, d, o\rangle \mid \langle et_i, et_j, d, o\rangle \in epo(ac), d \in C\}|}{\sum_{ac_k \in \mathbf{A}} |\{\langle et_i, et_j, d, o\rangle \mid \langle et_i, et_j, d, o\rangle \in epo(ac_k), d \in C\}|}$$

Intuitively, the confidence of a TP feedback item $\langle etp = \langle et_i, et_j, C\rangle, ac\rangle$ represents the conditional probability that the class of the current activity is a given the observation of two events $ev(et_i, t_i)$, $ev(et_j, t_j)$ satisfying C.

For each TP feedback item we also compute the estimated **similarity** as the median value of the similarity between TP feedback items' origin and target:

$$s(etp = \langle et_i, et_j, C\rangle, ac, h) = \operatorname*{median}_{\langle et_i, et_j, d, o\rangle \in epo(ac)} sim(h, o).$$

Finally, the system communicates to the target home each personalized TP feedback item:

$$\langle etp, ac, p, s\rangle,$$

where etp is an event temporal pattern, ac is an activity class, p is the confidence value and s is the estimated similarity.

4.3.2. Semantic correlation updater

Periodically, each home receives an update from the cloud service consisting of a set **P** of *personalized SC feedback items*. The semantic correlation updater algorithm analyzes **P** along with the semantic correlations inferred by the ontology in order to refine SC values. We denote $OSC(et, ac)$ as the semantic correlation between et and ac computed by the ontology, while $SC(et, ac)$ is the one computed by our algorithm. U is the set of *unpredictive event types*:

$$U = \{et \mid predAct(et) = \emptyset\};$$

i.e., U contains all the event types that the current ontology does not consider predictive for any activity.

The pseudo-code of the semantic correlation updater algorithm is shown in Algorithm 1. At first, the algorithm initializes the current semantic correlations with the ones computed by the ontology. The set *newevents* is initialized to the empty set. Then, the algorithm iterates on each personalized SC feedback item $\langle et, ac, p, s\rangle$ contained in **P** in order to update the semantic correlations produced by the ontology. If et belongs to U, $SC(et, ac)$ is set to its predictiveness value

Algorithm 1: Semantic correlation updater

Input: A set of personalized SC feedback items $P = \{\langle et_1, ac_1, p_1, s_1\rangle, \langle et_2, ac_2, p_2, s_2\rangle, \dots\}$, semantic correlation function OSC computed by the ontology, and set U of unpredictive events
Output: Refined semantic correlation function SC

```
1: $SC \leftarrow OSC$
2: $newevents \leftarrow \emptyset$
3: for each $\langle et, ac, c, s\rangle \in \mathbf{P}$ do
4:     if $et \in U$ then
5:         $SC(et, ac) \leftarrow c$
6:         if $et \notin newevents$ then
7:             $newevents \leftarrow newevents \cup \{et\}$
8:             for each $ac_i \in \mathbf{A}$ s.t. $ac_i \neq ac$ do
9:                 $SC(et, ac_i) \leftarrow 0$
10:            end for
11:        end if
12:    else if $OSC(et, ac) = 0$ then
13:        $ac_{ont} \leftarrow$ an activity $ac_j \in \mathbf{A}$ s.t. $OSC(et, ac_j) > 0$
14:        $SC(et, ac_{ont}) \leftarrow \frac{SC(et, ac_{ont})}{1 + s \cdot SC(et, ac_{ont})}$
15:        $SC(et, ac) \leftarrow s \cdot SC(et, ac_{ont})$
16:        for each $ac_i \in \mathbf{A}$ do
17:            if $ac_i \neq a_{ont}$ and $ac_i \neq ac$ then
18:                $SC(et, ac_i) \leftarrow SC(et, ac_i) \cdot (1 - SC(et, ac))$
19:            end if
20:        end for
21:    end if
22:    for each $et \in E$ , $ac \in A$ do
23:        $SC(et, ac) \leftarrow SC(et, ac) + \frac{1 - \sum_{ac' \in A} SC(et, ac')}{|A|}$
24:    end for
25: end for
26: return $SC$
```

p. Moreover, if et is observed for the first time during the current iteration (i.e., if it is not yet part of the set *newevents*), the semantic correlation value $SC(et, ac_i)$ for any other activity class $ac_i \neq ac$ is initialized to 0, and et is added to the set of new events. Intuitively, since *unpredictive event types* have uniform semantic correlations for all the activities, they are usually queried more than other event types since they contribute most in increasing the entropy value. This makes the *predictiveness* values provided by the CLOUD SERVICE reliable to be used as semantic correlations for et, thus overriding the uniform semantic correlations inferred by the ontology.

In the case of $et \notin U$, we update the semantic correlations only if $SC(et, ac)$ is 0. Indeed, if $SC(et, ac) > 0$ (i.e., et is already predictive for the activity ac in the ontology), we do not update the semantic correlations since we consider the correlations provided by the ontology as reliable.

Instead, whenever a new semantic correlation between et and ac is discovered from a personalized SC feedback item and in the ontology et was predictive for an activity $ac_{ont} \neq ac$ but not for ac, it is necessary to correspondingly scale all the other semantic correlations regarding et so that $\sum_{ac_k \in \mathbf{A}} SC(et, ac_k) = 1$. Hence, we select an activity ac_{ont} which is correlated to et according to the ontology (i.e., such that $OSC(et, ac_{ont}) > 0$). If there are multiple activities which are correlated to et according the ontology, we select one at random since our SEMANTIC CORRELATION REASONER uniformly distributes semantic correlations among activities. Then we scale $SC(et, ac_{ont})$ considering the *estimated similarity* value s:

$$SC(et, ac_{ont}) := \frac{SC(et, ac_{ont})}{1 + s \cdot SC(et, ac_{ont})}$$

Since the event types for which the ontology already provided a semantic correlation are seldom queried, it is not reliable to use the predictiveness value to update the semantic correlations. This is why we use the *estimated similarity s* instead. The next step consists in updating $SC(et, ac)$:

$$SC(et, ac) := s \cdot SC(et, ac_{ont})$$

Then, we update the semantic correlations of all the remaining activities ac_j (such that $ac_j \neq ac_{ont}$ and $ac_j \neq ac$) in the following way:

$$SC(et, ac_j) := SC(et, ac_j) \cdot (1 - SC(et, ac)).$$

Finally, we normalize the SC values in order to ensure that SC is a probability distribution, as required by property (1) introduced in Section 3. In particular, it can be easily verified that, by construction of Algorithm 1, for each event type et, the following property holds: $0 \leq \sum_{ac \in A} SC(et, ac) \leq 1$. Hence, for each $et \in E$ and $ac \in A$, the algorithm normalizes SC values applying the following operation:

$$SC(et, ac) := SC(et, ac) + \frac{1 - \sum_{ac' \in A} SC(et, ac')}{|A|}.$$

Table 1
Results (**F$_1$** score) of the proposed activity recognition method with collaborative active learning compared to related work for recognizing interleaved activities.

Activity	Supervised machine learning [40]	Unsupervised probabilistic reasoning [16]	newNECTAR without AL	NECTAR [35]	newNECTAR with AL
ac_1	0.80	0.74	0.78	**0.82**	**0.82**
ac_2	0.87	0.84	0.85	0.87	**0.91**
ac_3	0.59	0.36	0.70	0.71	**0.75**
ac_4	0.52	0.49	0.67	**0.72**	**0.72**
ac_5	**0.88**	0.83	0.77	0.78	0.87
ac_6	0.85	0.67	0.89	0.89	**0.90**
ac_7	0.57	0.36	0.46	0.63	**0.70**
ac_8	0.84	0.69	0.71	0.82	**0.87**
$avg.$	0.74	0.70	0.73	0.78	**0.82**

After each update, the function $SC(et, ac)$ computed by our algorithm thus replaces $OSC(et, ac)$ for both the QUERY DECISION and MLN_{NC} ACTIVITY RECOGNITION layers.

4.3.3. Temporal pattern updater

The goal of this module is to transform each personalized TP feedback item received from the cloud service into a soft rule to be added to the MLN model. The actual formalism to encode TP feedback items into MLN rules has been explained in Section 4.1.3. Given a TP feedback item $\langle etp, ac, p, s\rangle$, the weight w of the corresponding soft rule depends on the **budget** $\beta > 0$ associated by the system to temporal patterns, on the item's confidence p and on the item's estimated similarity s:

$$w = \beta \cdot p \cdot s.$$

The budget is a system parameter that determines the influence of temporal patterns on the MLN model: the higher the budget, the strongest the influence of TP rules on the model. For the sake of this work, the optimal value for the budget is chosen experimentally.

5. Experimental evaluation

In order to evaluate our system, we use the well-known CASAS dataset [38,39]. This dataset includes eight high-level ADLs performed by 21 subjects in a smart-home. Several sensors were deployed to monitor movements, use of water and interaction with objects, doors, and drawers. During the data collection, one subject at a time was present in the smart-home environment. Each subject was instructed to perform the following ADLs: *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 activities were performed both in sequential and interleaved fashion, and their execution time and order were up to the subject. For a more detailed description, we refer the reader to the original publication concerning the floor plan of the flat, the sensor positions, and a more detailed description of the activities [39].

We used the CASAS dataset to simulate 21 apartments with identical sensing infrastructures but inhabited by different subjects. This setup resembles the one of a residence for elderly people consisting of several similar apartments. We fixed the similarity $sim(h_1, h_2)$ between each pair of apartments to 0.5, since the sensing infrastructures are identical (i.e., their similarity is 1), while the profiling of the subjects is unknown.

During a pre-processing phase, we removed from the dataset the occurrences of those motion sensors which we found out to be *noisy*; i.e., producing measurements essentially independent from the performed activities. Most noisy motion sensors were those placed in locations irrelevant for the activity recognition task. Other ones triggered too many events, possibly due to excessively high sensitivity or too wide coverage area. Hence, we kept motion sensor events from 7 devices only.[1]

We performed *leave-one-subject-out* cross validation. In each fold, *newNECTAR* collects segment feedback items from 20 subjects and uses them to (a) update semantic correlations and (b) discover temporal patterns for the remaining subject.

Table 1 summarizes our overall experimental results and shows that the application of our novel collaborative active learning method increases the recognition performance of about 9% over *newNECTAR* without active learning. Moreover, *newNECTAR* also outperforms our previous method NECTAR [35] by 4%, thus showing that mining temporal patterns from feedback can significantly improve activity recognition.

In order to compare *newNECTAR* with state-of-the-art techniques, we also implemented the supervised method proposed in [40] which relies on machine learning and time-based feature extraction. As machine learning algorithm we used Random Forest, since it is commonly used in activity recognition systems [2,41]. Fig. 3 shows how Random Forest performs with respect to other classifiers.

[1] Those sensors are identified as M02, M03, M04, M05, M13, M23, and M24 in the dataset.

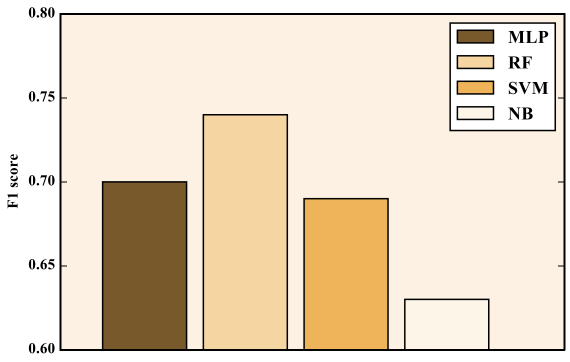


Fig. 3. The impact of different classifiers on the recognition rate. MLP = MultiLayer Perceptron, RF = Random Forest, SVM = Support Vector Machine, NB = Naive Bayes.

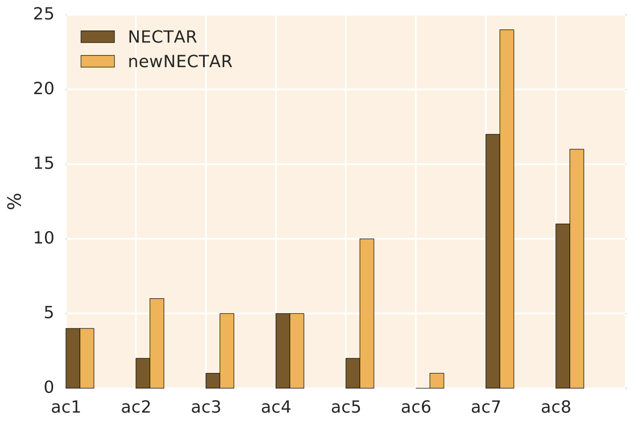


Fig. 4. Relative improvement (%) of using active learning considering both semantic correlations and temporal patterns (*newNECTAR*) with respect to considering only semantic correlations (NECTAR). $\lambda = 0.9$, $\sigma = 7.5$, $\sigma' = 0.01$, $\beta = 0.3$.

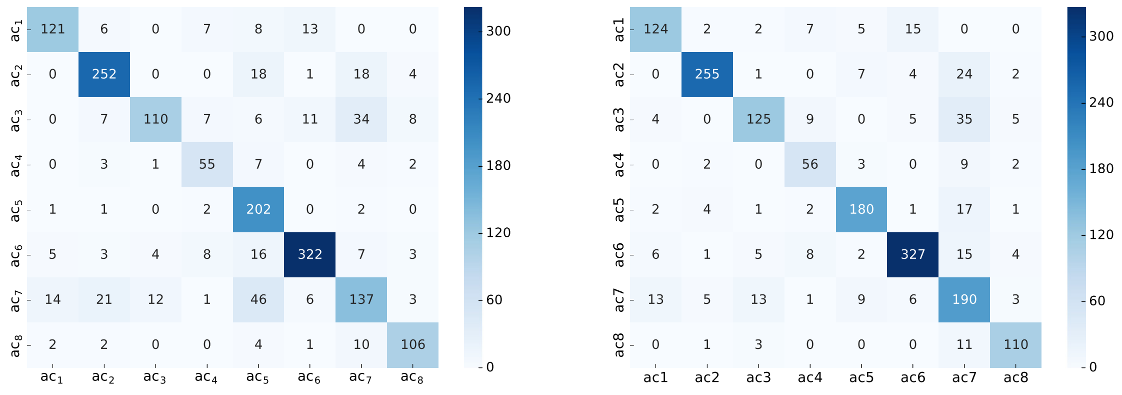
We executed the experiments using that method with the same dataset using leave-one-subject-out cross validation. Results show that *newNECTAR* outperforms the supervised method in terms of average F_1 (+8%). Individual activities are better or equal recognized, except for *prepare birthday card* (ac_5, −1%). We also compare *newNECTAR* with an unsupervised method which was recently proposed in [16]. That approach computes correlations between home infrastructure and activities by mining images scraped from the Web. Similarly to *newNECTAR*, those correlations are provided to a probabilistic reasoner in order to recognize activities on sensor data. Results show that *newNECTAR* outperforms that method by having a significant higher recognition rate (+12%). Moreover, each individual activity is better recognized by *newNECTAR*.

Inspecting the results of *newNECTAR* we observe that, with the introduction of active learning, the recognition rate always increases. In more detail, Fig. 4 shows that the recognition rate of *clean* has a significant increase (ac_7, +24%). On the other hand, *prepare soup* (ac_6) has a small improvement of only 1%. A deeper investigation pointed out that activity ac_6 was almost never queried, since its initial semantic correlations derived from our ontology were already sufficient to accurately recognize it. Regarding the other activities, we report an improvement which varies from 4% to 16%. With respect to our previous work [35], the recognition rate of the majority of the activities increased by considering temporal patterns. However, the activities *fill medication dispenser* (ac_1) and *answer phone* (ac_4) maintained the same recognition rate. A deeper investigation pointed out that our system almost never found relevant temporal patterns for those two activities.

Figs. 5a and 5b show respectively the confusion matrices for *NECTAR* and *newNECTAR*. This comparison shows that *newNECTAR* allows to obtain a significantly lower number of false positives.

For instance, *clean* (ac_7) is often confused with the remaining activities by *NECTAR*. This is due to the fact that *clean* is not clearly bound to a certain location or sensorized object; hence, during that activity the inhabitant triggers several sensor events that indicate the execution of other activities. On the other hand, *newNECTAR* considers reliable temporal patterns which allows to obtain a significant lower number of false positives for that activity, thus improving significantly the recognition rate.

Analyzing the diagonal, *newNECTAR* generates a smaller number of true positives for *prepare birthday card* (ac_5). However, considering *NECTAR*, that activity has a precision of 0.66 and a recall of 0.97, with an overall F1 score of 0.78; on



(a) NECTAR (b) *newNECTAR*

Fig. 5. Confusion matrix using our method. $\lambda = 0.9$, $\sigma = 7.5$, $\sigma' = 0.01$, $\beta = 0.3$.

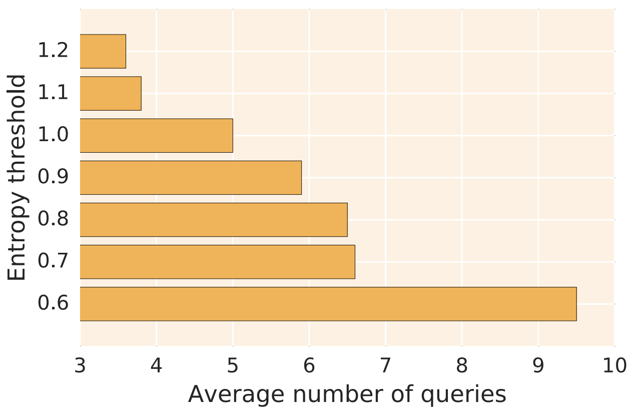


Fig. 6. How entropy affects the number of queries. $\sigma = 7.5$, $\sigma' = 0.01$, $\beta = 0.3$.

the other hand, *newNECTAR* eliminates a huge number of false positives, allowing to reach for both precision and recall metrics the value of 0.87.

The above mentioned results were obtained setting the entropy threshold to 0.9. As this value directly influences the number of queries issued by the system, it is an important parameter to consider. Fig. 6 clarifies that on average a user had to answer 6 questions to achieve the reported improvement of 9%. In the considered dataset, only one day of ADLs for each subject was available. We expect that the average number of queries in a day for a specific user will significantly decrease over time, thus converging to 0 queries after few days.

As Fig. 7 shows, lowering or increasing the entropy threshold leads to worse recognition rate. Increasing the entropy threshold means receiving less feedback from the participating inhabitants, and hence relying on less data to compute updates. On the other hand, asking more queries leads to introducing noisy data which makes update less accurate.

It is important to note that, without any active learning mechanism, the recognition rate is just 73%, which means that even with a high entropy threshold the improvement is significant (+7%). This is an indicator that useful information is part of queries with the highest entropy and that a less annoying system is still capable of significantly improve the recognition rate.

Besides entropy, we also assessed the impact of the support values σ (for semantic correlations) and σ' (for temporal patterns), which ensure the reliability of the updates. Fig. 8 outlines that when σ drops under a certain value, the system uses unreliable feedback, obtaining a detriment of recognition rates. On the contrary, using an excessively large value of σ, the system filters out relevant feedback that could improve recognition rates. We empirically determined that the optimal value of σ is $\sigma = 7.5$.

On the other hand, varying the support value σ' gives less intuitive outcomes, as Fig. 9 shows. Indeed, the optimal value $\sigma' = 0.002$ is very low. Moreover, a zero support value still gives nearly optimal results. This is due to how we assign weights to temporal rules in our MLN model. Since we assign to each temporal rule its confidence value scaled by the budget parameter β, temporal patterns with a very low confidence are associated with a very low weight in our MLN model. This makes unreliable temporal patterns only slightly influent in activity recognition. However, as Fig. 9b

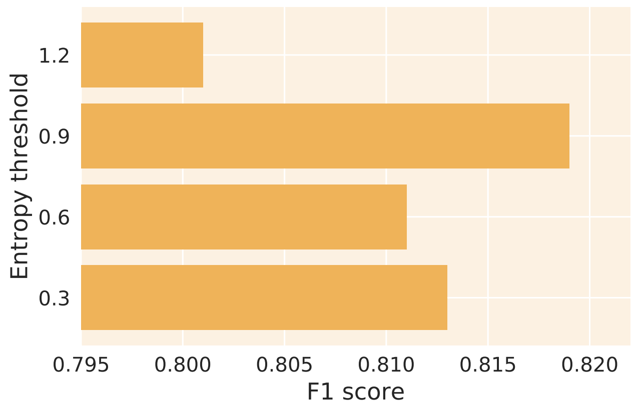


Fig. 7. How entropy affects F_1 score. $\sigma = 7.5$, $\sigma' = 0.01$, $\beta = 0.3$.

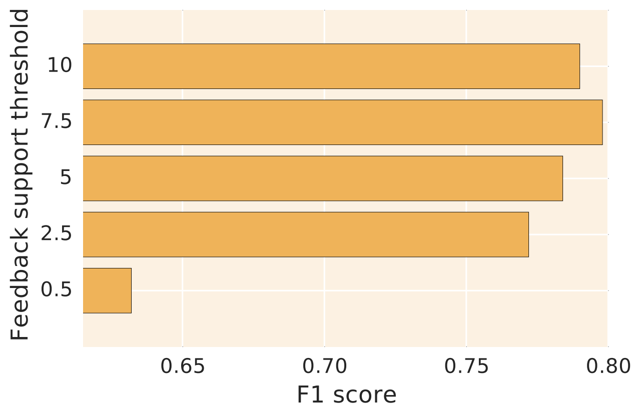


Fig. 8. How σ affects F_1. $\lambda = 0.9$, $\sigma' = 0.01$, $\beta = 0.3$.

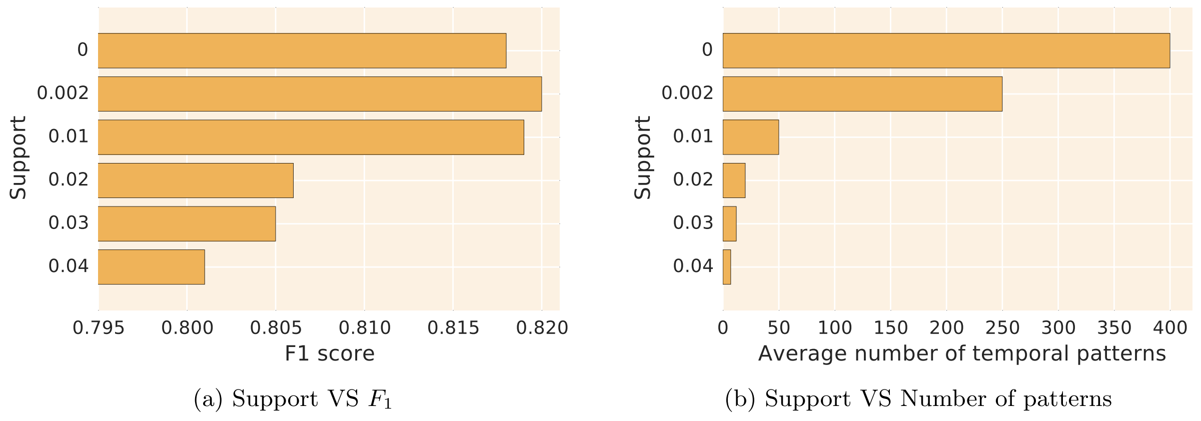


Fig. 9. How σ' impacts F1 score and the number of frequent pattern considered. $\lambda = 0.9$, $\sigma = 7.5$, $\beta = 0.3$.

shows, the main drawback is that $\sigma' = 0$ implies adding to the model an average of 400 temporal rules. This makes the activity recognition process very slow. The optimal value of $\sigma' = 0.002$ implies adding $\sim$ 250 temporal rules to our MLN model, still making the recognition process very slow. Instead, choosing $\sigma' = 0.01$ gives us a recognition rate which is worse only by 0.1%, with the advantage of a faster recognition process. Support values higher than 0.01 gives in general worse results, since we exclude temporal patterns which are important to obtain a good recognition rate. However, it is important to note that, if computational time is a priority, having high values of σ' still significantly improve results with respect to not having active learning at all.

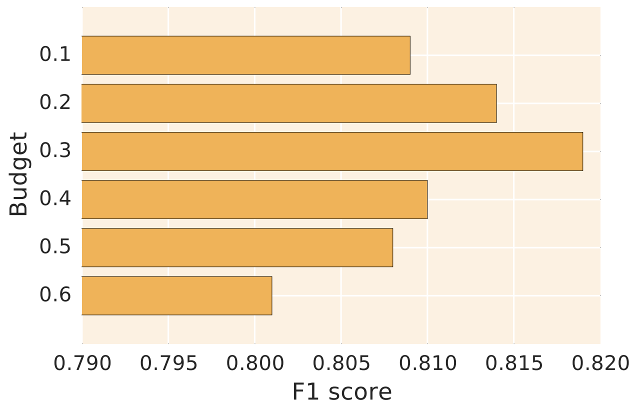


Fig. 10. How budget affects F_1 score. $\lambda = 0.9$, $\sigma = 7.5$, $\sigma' = 0.01$.

Finally, Fig. 10 shows how the budget parameter β (used to scale the weights of the MLN rules associated to temporal patterns) impacts on the recognition rate. Intuitively, high values of β lead to high weights which makes temporal patterns too prevalent with respect to the other rules of our MLN model. On the other hand, small values of β lead to small weights associated with temporal patterns, which consequently have less impact on activity recognition. The optimal value we evaluated is $\beta = 0.3$.

In general, our results clearly show that collaborative active learning is a reliable tool to discover new semantic correlations and temporal patterns, and at the same time to improve the recognition rate. This is especially the case for sensors that do not carry explicit semantic information with respect to activities. For instance, our ontology did not cover the events related to motion sensors. Our system was able to automatically learn the semantic correlation for those sensors' types improving the recognition rate.

Finally, we show results about the query rate (i.e., the percentage of queried segments) of *newNECTAR*. Since the CASAS dataset only considers one day of ADLs for each subject, we first computed the query rate for the 20 subjects that we used to collect feedback at each fold of our cross-validation. Those queries were triggered using the knowledge-based recognition model not yet affected by collaborative active learning. Since initially the model is inaccurate, it emerges that our system's query rate ranges between 17% and 43% (avg. 25%) depending on the subject. It is important to note that it is possible to trade a small amount of accuracy to significantly reduce the number of queries. For instance, increasing the entropy threshold to $\sigma = 1.2$, the query rate drops between 5% and 33% (avg. 21%) , with an overall F1 score which is only 1.8% lower than the one obtained with $\sigma = 0.9$. We expect that the query rate will decrease over time by incorporating feedback in the recognition model. In order to evaluate this aspect, we evaluated the impact of collaborative active learning on the number of queries triggered by the system. Hence, we computed the query rate for the subjects which we used as test set, whose recognition model has been corrected by collaborative active learning. Results show that our method allows to significantly decrease the query rate that drops between 0% and 20% (avg. 7%) considering entropy $\sigma = 1.2$. Unfortunately, due to the limited size of the dataset, we could not consider additional data to further evaluate the decrease of the query rate.

6. Discussion

6.1. Interaction with the inhabitant

In order to make our system practical in real scenarios, we aim to investigate important contextual aspects that should be considered when evaluating whether to ask a feedback or not. These aspects include the number of queries that have already been asked recently, the current mood of the subject and whether he/she can be currently interrupted. We aim to investigate if a game-theory based approach could be used to derive a personalized query policy capable to balance the need of the feedback and the estimated willingness of the subject to answer a query in a particular point in time.

Moreover, the interface used to query the user should be intuitive and user-friendly. We are developing a prototype of such interface, which also includes a speech recognition module in order to let the inhabitant answer queries in natural language. Voice interface is particularly suitable for elderly subjects, thus facilitating their interaction with our system. We will carry out extensive experiments to understand the impact of this interface in real scenarios.

6.2. Privacy aspects

For the sake of this work, we assumed that the CLOUD SERVICE is trusted, while in a real scenario it can be considered an untrusted *honest-but-curious* third party. Hence, there is the need of protecting the confidentiality and integrity of user and infrastructure profiles, as well as the information about events and activities provided by the feedback. We intend to investigate solutions based on homomorphic encryption [42] and secure multi-party computation [43] in order to let the CLOUD SERVICE run its algorithms on encrypted data.

6.3. Ontology engineering

Even if our system relies on a generic and possibly incomplete ontology which considers general relationships between activities and home infrastructure, the engineering effort is still noticeable. We believe that this effort could be reduced by re-using and extending existing ontologies. However, one could argue that it would be easier to manually estimate correlations among activities and sensor events based on common sense. However, manual modeling is unfeasible in realistic scenarios. For instance, the dataset we used in our experiments involves 70 sensors and 8 activities, resulting in 560 different values of semantic correlations. Other real-world deployments are much more complex. The collection and aggregation of feedback can also be exploited to revise the ontology with new correlations between event types and activities which were not considered in the first place. Hence, knowledge engineers could mine the collected feedback in order to decide whether and how to revise the ontology with new axioms.

7. Conclusion and future work

In this paper we presented a novel framework which exploits collaborative active learning to improve a generic ontological model of activities manually crafted by knowledge engineers. Experimental results show that our framework significantly improves the overall system's accuracy, while issuing a limited number of queries to the inhabitants. The current system re-evaluates the recognition model from scratch every time an update is received. As future work, we plan to improve it by devising an algorithm to continuously adjust correlations and temporal patterns as the updates are received.

Acknowledgments

The authors want to thank the anonymous reviewers' for their constructive feedback and comments to help improve the quality of this work. The work of D. Riboni was partially supported by the "DomuSafe" project funded by Sardinia regional government, Italy (CRP 69, L.R. 7 agosto 2007, n.7).

Conflict of interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] P.N. Dawadi, D.J. Cook, M. Schmitter-Edgecombe, Automated cognitive health assessment using smart home monitoring of complex tasks, IEEE Trans. Syst. Man Cybern. Syst. 43 (6) (2013) 1302–1313.
[2] D. Riboni, C. Bettini, G. Civitarese, Z.H. Janjua, R. Helaoui, Smartfaber: recognizing fine-grained abnormal behaviors for early detection of mild cognitive impairment, Artif. Intell. Med. 67 (2016) 57–74.
[3] H.M.S. Hossain, M.A.A.H. Khan, N. Roy, Active learning enabled activity recognition, Pervasive Mob. Comput. 38 (2017) 312–330.
[4] D. Riboni, T. Sztyler, G. Civitarese, H. Stuckenschmidt, Unsupervised recognition of interleaved activities of daily living through ontological and probabilistic reasoning, in: Proceedings of ACM UbiComp, ACM, 2016, pp. 1–12.
[5] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, H. Sagha, H. Bayati, M. Creatura, J. del R. Millán, Collecting complex activity datasets in highly rich networked sensor environments, in: Proceedings of the Seventh International Conference on Networked Sensing Systems, IEEE, 2010, pp. 233–240.
[6] S.W. Loke, Representing and reasoning with situations for context-aware pervasive computing: A logic programming perspective, Knowl. Eng. Rev. 19 (3) (2004) 213–233.
[7] X.H. Wang, T. Gu, D.Q. Zhang, H.K. Pung, Ontology based context modeling and reasoning using owl, in: Proceedings of Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, IEEE Computer Society, Washington, D.C., 2004, pp. 18–22.
[8] L. Chen, C.D. Nugent, Ontology-based activity recognition in intelligent pervasive environments, Int. J. Web Inf. Syst. 5 (4) (2009) 410–430.
[9] D.H. Hu, Q. Yang, Transfer learning for activity recognition via sensor mapping, in: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011, IJCAI/AAAI, 2011, pp. 1962–1967.
[10] D.J. Cook, K.D. Feuz, N.C. Krishnan, Transfer learning for activity recognition: A survey, Knowl. Inf. Syst. 36 (3) (2013) 537–556.
[11] D. Riboni, C. Bettini, OWL 2 Modeling and reasoning with complex human activities, Pervasive Mob. Comput. 7 (3) (2011) 379–395.
[12] R. Helaoui, D. Riboni, H. Stuckenschmidt, A probabilistic ontological framework for the recognition of multilevel human activities, in: Proceedings of ACM UbiComp, ACM, 2013, pp. 345–354.
[13] M. Perkowitz, M. Philipose, K.P. Fishkin, D.J. Patterson, Mining models of human activities from the web, in: Proceedings of WWW Conference, ACM, 2004, pp. 573–582.
[14] D. Wyatt, M. Philipose, T. Choudhury, Unsupervised activity recognition using automatically mined common sense, in: Proceedings AAAI, AAAI Press / The MIT Press, 2005, pp. 21–27.
[15] E.M. Tapia, T. Choudhury, M. Philipose, Building reliable activity models using hierarchical shrinkage and mined ontology, in: Proceedings of Pervasive, in: LNCS, vol. 3968, Springer, 2006, pp. 17–32.

[16] D. Riboni, M. Murtas, Web mining & computer vision: new partners for object-based activity recognition, in: Proceedings of the 26th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE Computer Society, 2017, pp. 158–163.
[17] A. Vahdatpour, N. Amini, M. Sarrafzadeh, Toward unsupervised activity discovery using multi-dimensional motif detection in time series, in: C. Boutilier (Ed.), Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI), 2009, pp. 1261–1266.
[18] P. Rashidi, D.J. Cook, L.B. Holder, M. Schmitter-Edgecombe, Discovering activities to recognize and track in a smart environment, IEEE Trans. Knowl. Data Eng. 23 (4) (2011) 527–539.
[19] E. Hoque, J.A. Stankovic, AALO: Activity recognition in smart homes using active learning in the presence of overlapped activities, in: Proceedings of the 6th PervasiveHealth Conference, IEEE, 2012, pp. 139–146.
[20] J. Ye, G. Stevenson, S. Dobson, Usmart: An unsupervised semantic mining activity recognition technique, ACM Trans. Interact. Intell. Syst. (TiiS) 4 (4) (2014) 16:1–16:27.
[21] R. Liu, T. Chen, L. Huang, Research on human activity recognition based on active learning, in: Machine Learning and Cybernetics (ICMLC), 2010 International Conference on, Vol. 1, IEEE, 2010, pp. 285–290.
[22] L. Yao, F. Nie, Q.Z. Sheng, T. Gu, X. Li, S. Wang, Learning from less for better: semi-supervised activity recognition via shared structure discovery, in: Proceedings of ACM UbiComp, ACM, 2016, pp. 13–24.
[23] T. Sztyler, H. Stuckenschmidt, Online personalization of cross-subjects based activity recognition models on wearable devices, in: Proceedings of IEEE PerCom, IEEE, 2017, pp. 180–189.
[24] M. Stikic, K.V. Laerhoven, B. Schiele, Exploring semi-supervised and active learning for activity recognition, in: Proceedings of the 12th IEEE International Symposium on Wearable Computers (ISWC), IEEE Computer Society, 2008, pp. 81–88.
[25] Y. Ho, C. Lu, I. Chen, S. Huang, C. Wang, L. Fu, et al., Active-learning assisted self-reconfigurable activity recognition in a dynamic environment, in: Proceedings of the 2009 IEEE international conference on Robotics and Automation, IEEE Press, 2009, pp. 1567–1572.
[26] L. Zhao, G. Sukthankar, R. Sukthankar, Robust active learning using crowdsourced annotations for activity recognition, in: Human Computation, Papers from the 2011 AAAI Workshop, San Francisco, California, USA, August 8, 2011, AAAI Press, 2011, pp. 74–79.
[27] W.S. Lasecki, Y.C. Song, H.A. Kautz, J.P. Bigham, Real-time crowd labeling for deployable activity recognition, in: Proceedings of Computer Supported Cooperative Work (CSCW), ACM, 2013, pp. 1203–1212.
[28] J. Ye, Slearn: Shared learning human activity labels across multiple datasets, in: 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE, 2018, pp. 125–134.
[29] J. Wen, J. Indulska, M. Zhong, Adaptive activity learning with dynamically available context, in: Proceedings of IEEE PerCom, IEEE Computer Society, 2016, pp. 1–11.
[30] J.F. Roddick, M. Spiliopoulou, A survey of temporal knowledge discovery paradigms and methods, IEEE Trans. Knowl. Data Eng. 14 (4) (2002) 750–767.
[31] K. Romer, Distributed mining of spatio-temporal event patterns in sensor networks, EAWMS/DCOSS 1 (1) (2006) 103–116.
[32] T. Gu, L. Wang, Z. Wu, X. Tao, J. Lu, A pattern mining approach to sensor-based human activity recognition, IEEE Trans. Knowl. Data Eng. 23 (9) (2011) 1359–1372.
[33] D. Lymberopoulos, A. Bamis, A. Savvides, Extracting spatiotemporal human activity patterns in assisted living using a home sensor network, Univ. Access Inf. Soc. 10 (2) (2011) 125–138.
[34] P. Rashidi, D.J. Cook, Mining sensor streams for discovering human activity patterns over time, in: Data Mining (ICDM), 2010 IEEE 10th International Conference on, IEEE, 2010, pp. 431–440.
[35] G. Civitarese, C. Bettini, T. Sztyler, D. Riboni, H. Stuckenschmidt, Nectar: knowledge-based collaborative active learning for activity recognition, in: 2018 IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE Computer Society, 2018, pp. 125–134.
[36] M. Chekol, J. Huber, C. Meilicke, H. Stuckenschmidt, Markov logic networks with numerical constraints, in: 22st European Conference on Artificial Intelligence (ECAI2016), IOS Press, Amsterdam, The Netherlands, 2016, pp. 1–9.
[37] M. Richardson, P. Domingos, Markov logic networks, Mach. Learn. 62 (2006) 107–136.
[38] G. Singla, D.J. Cook, M. Schmitter-Edgecombe, Tracking activities in complex settings using smart environment technologies, Int. J. Biosci. Psychiatr. Technol. 1 (1) (2009) 25–35.
[39] D.J. Cook, A.S. Crandall, B.L. Thomas, N.C. Krishnan, CASAS: a smart home in a box, Computer 46 (7) (2013) 62–69.
[40] N.C. Krishnan, D.J. Cook, Activity recognition on streaming sensor data, Pervasive Mob. Comput. 10 (2014) 138–154.
[41] T. Sztyler, H. Stuckenschmidt, On-body localization of wearable devices: An investigation of position-aware activity recognition, in: 2016 IEEE International Conference on Pervasive Computing and Communications (PerCom), IEEE Computer Society, Washington, D.C., 2016, pp. 1–9.
[42] A.A. Atayero, O. Feyisetan, Security issues in cloud computing: the potentials of homomorphic encryption, J. Emerging Trends Comput. Inf. Sci. 2 (10) (2011) 546–552.
[43] Y. Lindell, B. Pinkas, Secure multiparty computation for privacy-preserving data mining, J. Priv. Confidentiality 1 (1) (2009) 5.