Improving activity classification using ontologies to expand features in smart environments

Alberto Salguero[1] and Macarena Espinilla[2]

[1] alberto.salguero@uca.es Universidad de Cádiz, Cádiz, Spain
[2] mestevez@ujaen.es Universidad de Jaén, Jaén, Spain

Abstract. Activity recognition is a promising field of research aiming to develop solutions within smart environments to provide relevant solutions on ambient assisted living, among others. The process of activity recognition aims to recognize the actions and goals of one or more person in a environment with a set of sensors are deployed, basing on the sensor data stream that capture a series of observations of actions and environmental conditions. This contributions presents the initial results from a new methodology that considers the use of ontologies to expand the set of feature vector, which is computed by using the sensor data stream, that is used in the process of activity recognition by data-driven approaches. The obtained results indicates that the use of extended feature vectors provided by the use of ontology offers a better accuracy regarding the original feature vectors used in the process of activity recognition with different data-driven approaches.

Keywords: Activity recognition; Smart environments; Ontology; Data-Driven approaches; Knowledge-Driven approaches

1 Introduction

Sensor-based activity recognition [2] is a very relevant process at the core of smart environments. This type of activity recognition is focused on recognizing the actions of one or more persons within the smart environment based on a series of observations of sub-actions and environmental conditions over a period of finite time. It can be deemed as a complex process that involves the following steps: i) select and deploy the appropriate sensors to be attached to objects within the smart environment; ii) collect, store and pre-process the sensor related data and, finally, iii) to classify activities from the sensor data through the use of activity models.

The sensor-based activity recognition is particular suitable to deal with activities that involve a number of objects within an environment, or instrumental activities of daily living (ADL)[9]. Approaches used for sensor-based activity

recognition have been divided into two main categories: Data-Driven (DDA) and Knowledge-Driven (KDA) approaches [2].

The former, DDA, are based on machine learning techniques in which a preexistent dataset of user behaviors is required. A training process is carried out, usually, to build an activity model which is followed by a testing processes to evaluate the generalization of the model in classifying unseen activities [10]. The advantages of the DDA are the capabilities of handling uncertainty and temporal information. However, these approaches require large datasets for training and learning, and suffer from the data scarcity or the cold start problem.

There are repositories which contain several ADL datasets in smart environments. One of the most well-known repositories is CASAS[3] [6]. In the context, it is interesting to mention the Open Data Initiative (ODI) [11] for Activity Recognition consortium that aims to create a structured approach to provide annotated datasets in an accessible format.

With KDA, an activity model is built through the incorporation of rich prior domain knowledge gleaned from the application domain, using knowledge engineering and knowledge management techniques [3,5]. KDA has the advantages of being semantically clear, logically elegant, and easy to get started. Nonetheless, they are weak to deal with uncertainty and temporal information as well as the activity models can be considered as static and incomplete.

In the context of KDAs, ontologies for activity recognition have provided success results. In this kind of approach, interpretable activity models are built in order to match different object names with a term in an ontology that is related to a particular activity.

Some hybrid approaches have been developed [4,12] that take advantage of the main benefits provided by DDAs and the use of an ontology. Thereby, ontological ADL models capture and encode rich domain knowledge and heuristics in a machine understandable and processable way.

This contribution of our current work proposes to use an ontology in order to extend the feature vectors to enrich these vectors through inferred knowledge in the ontology, improving the accuracy of classifiers based on DDAs used in the recognition of activities against the unextended feature vectors.

An evaluation is undertaken with a popular dataset to consider the effects of the extension of feature vectors in terms of overall accuracy for activity recognition based on sensor data gleaned from smart environments.

The remainder of the paper is structured as follows: Section 2 reviews some notions about ontologies that are needed to understand our proposal. Section 3 proposes the methodology to extended the set of feature vectors by means of an ontology. Section 4 presents an empirical study that analyzes our proposed methodology of extended feature vector in terms of accuracy based on a popular dataset by using the ontology. Finally, in Section 5, conclusions and future work are presented.

[3] http://ailab.wsu.edu/casas/datasets/ (last checked on April 19, 2017)

Table 1. Semantic of OWL logical operators

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

2 Review of ontologies

In this section some relevant concepts related to ontologies are reviewed in order to understand our proposed methodology. Ontologies are used to provide structured vocabularies that explain the relations among terms, allowing an unambiguous interpretation of their meaning. Ontologies are formed by concepts (or classes) which are, usually, organized in hierarchies [1,14], being the ontologies more complex than taxonomies because they not only consider *type-of* relations, but they also consider other relations, including *part-of* or domain-specific relations [8].

A formal language for working with ontologies is OWL [7,13], which is developed by the World Wide Web Consortium (W3C). The design of OWL is greatly influenced by Description Logics (DL), particularly in the formalism of semantics, the choice of language constructs and the integration of data types and data values.

In an ontology, the symbol $\top$ stands for the *top* concept of the hierarchy, all being concepts subsets of $\top$. The *subsumption* relation is usually expressed using the symbol $A \sqsubseteq B$, meaning that the concept A is a subset of the concept B. Concepts can also be specified as logical combinations of other concepts. The semantic of operators for combining concepts is shown in Table 1, where $C, C_1, C_2 \sqsubseteq \top$, R is a relation among concepts, Δ_I is the domain of individuals in the model and I is an interpretation function.

3 Methodology

This section describes the proposed methodology, which is made up of several independent applications. The purpose of the methodology is to add relevant features to the dataset to improve the accuracy of the classifiers in DDA by means an ontology.

3.1 An ontology for the description of activities

To describe the activities in [15] an ontology has been developed in OWL. The ontology defines two basic concepts, *Activity* and *Event*, which respectively rep-

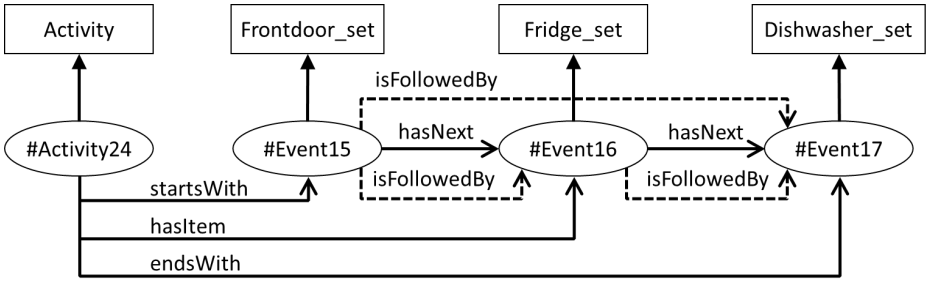

Fig. 1. Ontology example.

resent all the activities in the dataset and the activation of the sensors during these activities. Fourteen new subclasses of the *Event* class have been defined, each of them representing the activation during the activity of each of the sensors in the dataset[4]. The class *Frontdoor_set*, for example, represents the set of events corresponding to the activation of the sensor in the front door.

To relate sensors events to the activities we have defined four properties, shown in Figure 1. The properties *startsWith* and *endsWith* relate a particular activity to the first and the last events that occur during that activity, respectively. Both properties have been defined as functional, since an activity can only begin and end with a unique event. They have been declared as sub-properties of the *hasItem* property, which relates an activity to the events that have occurred during that activity. The property *hasItem* has been defined as inverse functional, since an event may occurs just in one activity. The class description *hasItem some Frontdoor_set*, for example, represents all those activities during which the front door sensor has been fired.

The order among events produced in an activity is maintained through the *hasNext* property. This property has been defined as functional and inverse functional, since an event can only be immediately followed or preceded by a single event. It has also been declared as an asymmetric and irreflexive property, since an event that happens after another event cannot happen before the former one, nor after itself. This property allows us to describe activities as chains of events. The class description *startsWith some (Frondoor_set and hasNext some (Fridge_set and hasNext some Dishwasher_set))*, for example, represents the activities that begin with the activation of the sensor of the front door, which is immediately followed by the activation of the fridge sensor and then by the dishwasher sensor, immediately. The activity *#Activity24* in Figure 1 is an example of activity described by the above class description.

[4] For simplicity, only the activation of the sensors has been taken into account for the experiment in Section 4.2. However, it is also possible to consider the deactivations of the sensors by just enabling a flag in the application developed for loading the dataset in the ontology.

The *hasNext* property has been declared as a subproperty of the *isFollowedBy* transitive property, which relates an event to all events that happen after it in an activity. Events related through the latter property do not have to occur consecutively in the activity. The class description *hasItem some (Frontdoor_set and isFollowedBy some Dishwasher_set)* is another way of describing the activity *#Activity24* of the example in Figure 1.

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

3.2 Extended features generation

In this section we explain how the features are generated from the information in the ontology. Basically, the idea consists on the combination of the entities in the ontology (concepts and relations), by means of logic operators, to generate new class descriptions that may be useful for the classification of the activities. Eventually, a class description that describes certain kind of activities may be found and selected as a new feature for the classifiers. The process is repeated until sufficient number of new relevant features are found. All the components of the methodology proposed in this work and how the information flows among them are explained in this section.

The system starts from a dataset with a set of labeled activities. First of all it is necessary to convert the dataset information into an ontology. In the experiment described in Section 4.2 an application has been developed to convert the information in [15] into an ontology following the rules in Section 3.1. In this first step two text files are also generated that contain: a) the list of individuals in the ontology of the kind of activity to be recognized by the classifier (positives), and b) the rest of individuals (negatives). These lists of individuals will be used to generate the input data for the classifier in a later step.

Next, it is necessary to expand the definition of the *Activity* concept in the ontology. The expansion process consists on generating new class descriptions that represent different patterns of activities, without taking into account the specific type of activity that is going to be recognized by the classifier. More specifically, the *OWLExpand* program takes the concept to be expanded (*Activity*) as an argument and uses a given set of classes, properties and operators to construct new class descriptions in DL. All the concepts in the ontology that at least describe some individual in the ontology have been taken as the set of classes L. All properties defined in the ontology have been taken as the set of properties P. The set of operators O is specified by the user and consists on a subset of all operators that can be used to combine class descriptions (see Section 2).

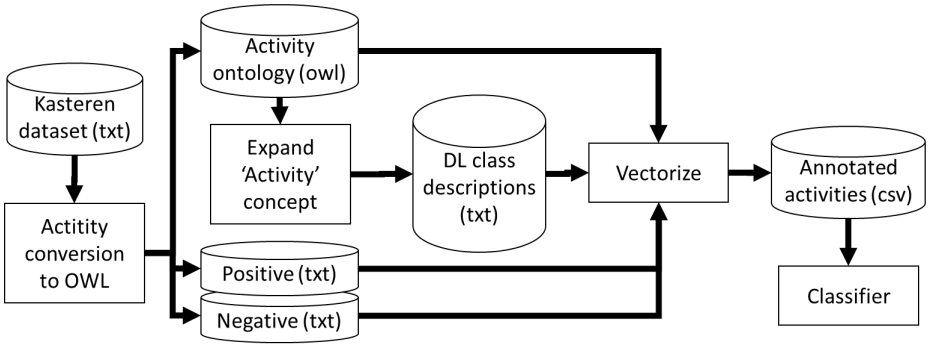

Fig. 2. Functional architecture.

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

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

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

The last type of operator that implements the *OWLExpand* application is the cardinality constraints. These operators limit the number of individuals to which an individual may be related among a given property. The class descriptions generated with this operators have the form $p_i\ or_k\ n\ c_j$, where $p_i \in P$, $o_k \in \{min, max, exact\}$, $c_j \in L$ and $n \in N$. Expressions such as *isFollowedBy min 4 Event* or *isFollowedBy exact 2 Cupboard_set* are generated, for example, representing the set of individuals followed by at least four sensor activations and the set of individuals followed by exactly two activations of the *Cupboard* sensor, respectively. The number of constraints to be generated is virtually infinite since $n \in N$. It is the user who must specify the values for n. For example, $n \in \{2, 3\}$ in the experiment of Section 4.2.

All the expressions generated are added to L and the process is repeated again. However, not all of the expressions generated are relevant. Some of them

Table 2. Example of resultant data

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

are simply unsatisfiable. A class expression such as *hasItem some Activity*, for example, is unsatisfiable since the range of the property *hasItem* is the *Event* concept, which is defined to be disjoint with the concept *Activity*. There cannot be an individual in the ontology which meets such restriction. For the same reason, expressions like *hasItem some startsWith some HallBedroom_Door_set* are also unsatisfiable, since the domain of the *startsWith* property is the concept *Activity*. Only satisfiable class expressions are added to L.

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

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

4 Experiment

In this contribution, a popular activity recognition dataset [15] of a a smart environment is used to evaluate the performance of our proposal. In this section we first describe the dataset and the experiment. Then, we compare the results obtained by classifiers using the classical approach and classifiers using the methodology proposed in this work.

4.1 From the sensor data stream to feature vectors

The dataset [15] used in the experiment to evaluate our proposal is composed by binary temporal data from a number of sensors, which monitored the ADLs

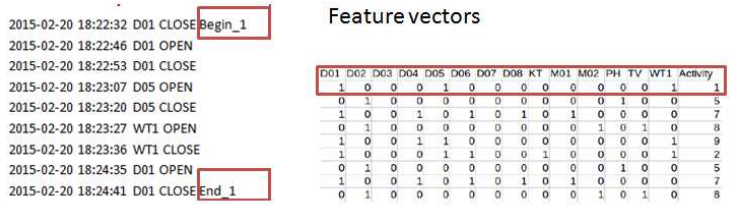

Fig. 3. Partial sensor data stream of a dataset and its computed feature vector

carried out in a home setting by a single inhabitant. This dataset was collected in the house of a 26-year-old male who lived alone in a three-room apartment. This dataset contains 245 activities that are annotated in the stream of state-change sensors generated by 14 binary sensors.

Each sensor is located in one of 14 different places within a home setting: microwave, hall-toilet door, hall-bathroom door, cups' cupboard, fridge, plates' cupboard, front door, dishwasher, toilet flush, freezer, pans' cupboard, washing machine, groceries' cupboard and hall-bedroom door. Sensors were left unattended, collecting data for 28 days in the apartment. Activities were annotated by the subject himself using a Bluetooth headset.

Seven different activities are annotated, namely: going to bed, using toilet, preparing breakfast, preparing dinner, getting a drink, taking a shower and leaving the house.

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

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

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

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

4.2 Experiment description

In order to evaluate the quality of the methodology proposed in this work an experiment has been carried out, in which the dataset in [15] have been used. The objective of the experiment is to determine whether or not a particular activity has been performed based on the sensors that have been fired during a specific period of time. To simplify the experiment, the time intervals always correspond to the annotated activities in the dataset.

The results obtained by four classifiers that use a classic DDA to solve this problem have been taken as reference to measure the efficiency of our proposal. For this purpose an application that identifies the sensors that have been fired during each of the activities has been built. The application generates a file in Weka format, following the structure presented in Section 4.1. This file contains an instance for each activity and as many features as sensors in the dataset. All the features are binary and specify if the sensor has been fired during the activity or not. Finally, it includes a class attribute, also binary, that indicates if it is the activity that the classifier is learning to identify or not. Each experiment consists, therefore, in determining which combination of sensors are fired for a particular activity, such as 'take shower', for example.

By using the Weka data mining software, we have generated C4.5[5], Sequential Minimal Optimization (SMO), Voted perceptron (VP) and Decision Table (DT) classifiers for all the activities in the dataset. The most difficult activities to be identified are 'go to bed' and 'use toilet', with 94.67% and 91.97% accuracy, respectively. These are the activities chosen to test the performance of the proposed system in this work.

Starting from the ontology proposed in Section 3.1, to which the information contained in [15] has been added, the *OWLExpand* program is used to generate new class descriptions automatically. All new class descriptions describe the *Activity* concept. Three subsets of operators have been used to generate three different sets of new class descriptions. For the first one, all available operators ($\mathcal{ACIXMSU}$) have been used. The complement, minimum cardinality and the existential quantifier operators ($\mathcal{CMS}$) have been used for the second one and only the existential quantifier ($\mathcal{S}$) has been used for the latter. In addition, versions with 50, 100, 150 and 200 class expressions have been generated for each of these sets.

All files with new class descriptions are evaluated by the *OWLVectorize* application and a new file in Weka format is generated for each of them. The same four types of classifiers that were employed to evaluate the performance of the classifiers using the classical approach have been used to evaluate the accuracy of the classifiers based on the new approach. Results are discussed in the next section.

4.3 Results

The accuracy obtained by all the different classifiers generated in the previous section for the 'go to bed' and 'use toilet' activities are shown in the Table 3 and Table 4, respectively. The first column indicates the approach used to generate the $|\mathcal{F}|$ features for the classifier, in the second column. The remaining columns indicate the accuracy of the corresponding classifier, in percentage values.

The first row contains the data related to the classifiers constructed using the classical approach. This approach yields a precision of 94.67 % for the best case of the activity 'go to bed'. In spite of being a very high value, most of the

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

Table 3. 'Go to bed' classification accuracy

Dataset	$\|\mathcal{F}\|$	C4.5	SMO	VP	DT
Classic	14	93,87	94,67	91,72	94,67
$\mathcal{ACIXMSU}$	50	90,65	91,04	90,37	90,23
$\mathcal{ACIXMSU}$	100	98,66	98,51	96,48	**98,79**
$\mathcal{ACIXMSU}$	150	98,66	98,78	97,02	98,79
$\mathcal{ACIXMSU}$	200	98,66	98,37	93,63	98,79
$\mathcal{CMS}$	50	90,65	91,04	90,37	90,23
$\mathcal{CMS}$	100	98,24	**99,06**	96,06	98,38
$\mathcal{CMS}$	150	98,24	98,38	94,16	98,38
$\mathcal{CMS}$	200	98,24	98,38	92,81	98,38
$\mathcal{S}$	50	98,52	98,38	93,88	98,79
$\mathcal{S}$	100	98,52	**98,92**	92,13	98,79
$\mathcal{S}$	150	98,11	98,24	96,21	98,11
$\mathcal{S}$	200	98,11	98,38	96,73	98,11
$\mathcal{S}$	300	98,11	98,24	94,70	98,11
$\mathcal{S}$	400	98,11	98,38	94,58	98,11
$\mathcal{S}$	500	98,11	98,66	95,39	98,11

Table 4. 'Use toilet' classification accuracy

Dataset	$\|\mathcal{F}\|$	C4.5	SMO	VP	DT
Classic	14	91,97	91,16	89,39	90,09
$\mathcal{ACIXMSU}$	50	86,53	87,33	84,88	85,03
$\mathcal{ACIXMSU}$	100	95,77	**96,86**	93,72	94,8
$\mathcal{ACIXMSU}$	150	95,77	96,16	92,9	94,8
$\mathcal{ACIXMSU}$	200	95,77	96,3	91,94	95,07
$\mathcal{CMS}$	50	86,53	87,33	84,88	85,03
$\mathcal{CMS}$	100	95,77	95,76	90,33	94,53
$\mathcal{CMS}$	150	95,22	96,16	90,19	93,71
$\mathcal{CMS}$	200	95,22	**96,29**	89,79	93,71
$\mathcal{S}$	50	96,32	96,32	91,28	95,76
$\mathcal{S}$	100	95,62	**97,67**	90,87	94,81
$\mathcal{S}$	150	97,26	97,53	92,76	94,96
$\mathcal{S}$	200	97,26	97,26	91,55	94,96
$\mathcal{S}$	300	97,26	97,53	90,59	94,96
$\mathcal{S}$	400	97,26	96,72	90,46	94,96
$\mathcal{S}$	500	97,26	96,45	89,53	94,96

classifiers created with the approach proposed in this work surpassed that value. The worst results are obtained for classifiers using the new approach, but just when the number of features (class descriptions) generated is insufficient.

Regarding the accuracy of the classifiers with respect to the set of operators used, it is noteworthy that there is no significant difference. The best values obtained for $\mathcal{ACIXMSU}$, $\mathcal{CMS}$ and $\mathcal{S}$ have been 98.79, 99.06 and 98.92 %, respectively. All these values have been obtained for one hundred or more features.

However, there is a significant difference in the time required to evaluate the class descriptions generated from the different sets of operators. The total time required for the generation and evaluation of the different characteristics varies between 2.07 and 1577.28 seconds. The best precision for the activity 'go to bed' is obtained for the classifier $\mathcal{CMS}$, using a total of 233.84 seconds to obtain an accuracy of 99.06 %. However, it takes only 2.60 seconds to get an accuracy of 98.92 % for the classifier that only uses the existential quantifier as the operator to generate the class expressions ($\mathcal{S}$).

The class expressions generated by the $\mathcal{S}$ classifier include, for example,

$$\textit{hasItem some (isFollowedBy some Hall-Toilet\_door\_set)} \quad (1)$$

$$\textit{startsWith some (hasNext some Hall-Bathroom\_door\_set)} \quad (2)$$

which represent activities in which there is a sensor that is fired before the *Hall-Toilet_door* sensor (1) and activities in which the *Hall-Bathroom_door* is the second sensor that has been fired (2).

A very similar behavior is observed in the case of the activity 'use toilet'. As shown in Table 4, the accuracy of the classifiers using the methodology proposed in this work increases significantly when one hundred or more features are generated. It goes from an accuracy of 91.92 % with the best case of *DT* to an accuracy of 97.67 % when only the existential quantifier ($\mathcal{S}$) is used as the operator to generate new class expressions.

Table 5. Global classification accuracy

Activity	Classic	Proposal	Gain	% Gain
Go to bed	94.67	99.06	4.39	82.36
Prepare dinner	97.68	99.60	1.92	82.76
Take shower	97.96	99.59	1.63	79.90
Use toilet	91.97	97.67	5.70	70.98
Average	95.57	98.98	3.41	79.00

Finally, it should be noted that the classifier based on neural networks (VP) is the classifier that worst responded to the feature expansion process proposed in this work. The improvement in the case of the activity 'use toilet' is insignificant. In any case, one of the advantages offered by the system proposed in this work is the possibility of identifying the relevant features for the classifiers. However, the classifier based on neural networks is the only classifier of the four employed in the test that does not allow the identification of these features, because it is a 'black box' classifier.

Table 5 summarizes the results discussed above and also includes the results obtained for other two activities in the dataset. Again, we have selected the other two activities in the dataset for which the classifiers using the classic approach obtain worst results, shown in the second column. It is worth to note the so high accuracy values that the classifiers using the classic approach obtain for these two new activities selected. The third column shows the accuracy of the best classifier that uses the proposed approach. The difference between those values is shown in fifth column and, finally, the last column indicates the percentage of gain achieved by the classifier with respect to the maximum possible gain. As it can be seen, the classifiers based on the proposal presented in this paper improve the accuracy obtained by the classifiers using the classic approach in all cases. The average classification accuracy for the classifiers using the classic approach is 95.57. The average classification accuracy for the classifiers using the proposed approach is 98.98. This mean that the classifiers using the proposed approach have trimmed down the difference with respect to the perfect classifiers a 79.00%.

5 Conclusion and future work

This contribution has been focused on a new methodology that uses ontology for the purpose of binary sensor-based activity recognition with DDA in order to increase the accuracy in the classification process when a single activity is carried out by a single person. To do so, the set of feature vector computed by the dataset are extended with the inferred knowledge from the ontology. An evaluation has been carried out with the following four popular classifiers: C4.5, Sequential Minimal Optimization, Voted perceptron and Decision Table. Results from the evaluation demonstrated the ability of the ontology to extend the vector

features to provide an increase of the performance in all evaluated classifiers. Our future work is addressed on evaluating the proposed methodology by means a dataset with a greater number of sensors and activities.

Acknowledgments

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

References

1. Chandrasekaran, B., Josephson, J., Benjamins, V.: What are ontologies, and why do we need them? IEEE Intelligent Systems and Their Applications 14(1), 20–26 (1999)
2. Chen, L., Hoey, J., Nugent, C., Cook, D., Yu, Z.: Sensor-based activity recognition. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 42(6), 790–808 (2012)
3. Chen, L., Nugent, C.: Ontology-based activity recognition in intelligent pervasive environments. International Journal of Web Information Systems 5(4), 410–430 (2009)
4. Chen, L., Nugent, C., Okeyo, G.: An ontology-based hybrid approach to activity modeling for smart homes. IEEE Transactions on Human-Machine Systems 44(1), 92–105 (2014)
5. Chen, L., Nugent, C., Wang, H.: A knowledge-driven approach to activity recognition in smart homes. IEEE Transactions on Knowledge and Data Engineering 24(6), 961–974 (2012)
6. Cook, D., Schmitter-Edgecombe, M., Crandall, A., Sanders, C., Thomas, B.: Collecting and disseminating smart home sensor data in the casas project. In: Proceedings of the CHI Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research (2009)
7. Horrocks, I., Patel-Schneider, P., Van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Web Semantics 1(1), 7–26 (2003)
8. Knijff, J., Frasincar, F., Hogenboom, F.: Domain taxonomy learning from text: The subsumption method versus hierarchical clustering. Data & Knowledge Engineering 83(0), 54–69 (2013)
9. Korhonen, I., Parkka, J., Van Gils, M.: Health monitoring in the home of the future. IEEE Engineering in Medicine and Biology Magazine 22(3), 66–73 (2003)
10. Li, C., Lin, M., Yang, L., Ding, C.: Integrating the enriched feature with machine learning algorithms for human movement and fall detection. Journal of Supercomputing 67(3), 854–865 (2014)
11. Nugent, C., Synnott, J., Santanna, A., Espinilla, M., Cleland, I., Banos, O., L.J., Hallberg, J., Calzada, A.: An initiative for the creation of open datasets within the pervasive healthcare. pp. 180–183 (2016)
12. Rafferty, J., Chen, L., Nugent, C., Liu, J.: Goal lifecycles and ontological models for intention based assistive living within smart environments. Computer Systems Science and Engineering 30(1), 7–18 (2015)

13. Sirin, E., Parsia, B., Grau, B., Kalyanpur, A., Katz, Y.: Pellet: A practical owl-dl reasoner. Web Semantics 5(2), 51–53 (2007)
14. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowledge Engineering Review 11(2), 93–136 (1996)
15. Van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B.: Accurate activity recognition in a home setting. pp. 1–9 (2008)