Advance Access publication on
doi: 10.1093/comjnl/bxaa179

Hybrid Learning Approach Toward Situation Recognition and Handling

Hossein Rajaby Faghihi, MohammadAmin Fazli* and Jafar Habibi

Department of Computer Engineering, Sharif University of Technology, Azadi Avenue, Tehran, Iran
**Corresponding author: fazli@sharif.edu*

We propose a novel hybrid learning approach to gain situation awareness in smart environments by introducing a new situation identifier that combines an expert system and a machine learning approach. Traditionally, expert systems and machine learning approaches have been widely used independently to detect ongoing situations as the main functionality in smart environments in various domains. Expert systems lack the functionality to adapt the system to each user and are expensive to design based on each setting. On the other hand, machine learning approaches fail in the challenge of cold start and making explainable decisions. Using both of these approaches enables the system to use user's feedback and capture environmental changes while exploiting the initial expert knowledge to solve the mentioned challenges. We use decision trees and situation templates as the core structure to interpret sensor data. To evaluate the proposed method, we generate a new human-annotated dataset simulating a smart environment. Our experiments show superior results compared with the initial expert system and the machine learning approach while preserving the initial expert system's interpretability.

Keywords: hybridlearning;situation awareness; internet of things; situation recognition.;

Received 1 September 2019; Revised 4 November 2020; Accepted 4 November 2020
Handling editor: Fabrizio Messina

1. INTRODUCTION

The boost of techniques in machine learning and pervasive computing and the increased accessibility of environmental data have resulted in an increasing interest in smart environments. Moreover, smart environments have shown promising results in various areas, such as healthcare monitoring, elderly assisting and surveillance. On the other side, the reduction of power consumption in electronic components and the spread of wireless communication technologies provide a feasible platform for smart environments to play an essential role in daily human life [1]. Furthermore, as sensors get more robust and varied, smart environments can become smarter due to the availability of comprehensive information from their surroundings.

Sensor data are useful in understanding the ongoing situation in an environment and in offering relevant services to the environment's users. However, these data are almost meaningless without an interpretation process. Smart environments can gain a better understanding of their surroundings by analyzing raw sensor data. This newfound understatement is essential to enable a productive interaction between smart environments and users. However, it is quite challenging due to the difficulties of interacting with a living agent.

The first exquisite challenge is the dynamic nature of such interaction. Interactions are not likely to follow a particular pattern during a long period and tend to change frequently. Changes can appear in the behavior of living agents or the availability and reliability of sensor data. As shown in HCI[1] research [2], the most important goal of situation-aware environments is helping their users to achieve their objectives by providing them with relevant services.

Another challenge is the considerable amount of data generated frequently by many sensors, making it hard to maintain a real-time interaction. Moreover, the generated data by sensors are often unreliable due to the possible corruption of communication tools or low accuracy measurements. Furthermore, considering only the recent data received from sensors is often inadequate to understand changes in an environment. Hence, storing historical data is a mandatory task. For example, it is not clear if a man is entering a room by looking at his presence in the room and his previous location is required to identify whether he has just entered the room or was already in there.

[1] Human–computer interaction

The process of detecting a situation in a smart environment consists of (i) gathering data from sensor networks, (ii) deriving more abstract information from sensor data and (iii) identifying occurring situation by considering the abstract information provided by the second step [3]. The abstract information derived from the sensor data is commonly referred to as context. Context is any information that can characterize an entity's situation, where the entity is an object playing a role in the environment [4].

The ability of a smart environment to be aware of context is called context awareness. It is formally defined as the use of context by the smart environment to provide task-relevant information or services to a user [5]. Context awareness is considered as a key factor in developing intelligent and practical systems and is considered important in many applications such as intelligent transportation [6–11], smart cities [12, 13] and healthcare [14–17].

The context requires further interpretation to yield a better understanding of the environment. This process is called situation identification. A situation is a more complex structure than context; the situation involves complex relations and temporal aspects with a need for additional semantic interpretation [18]. Situation awareness is defined as the understanding of factors in an environment regarding time and space, the appreciation of their semantic and a hypothetical status of what is going to happen to them [19].

Many methods and approaches have been proposed for achieving situation awareness in smart environments. Some methods use machine learning [20–25], while others utilize specification-based [2, 18, 26, 27] methods [28]. Recent approaches [29] combine both types of methods and tackle this issue with a hybrid setting. Hybrid methods tend to use the benefits of both approaches while eliminating their weaknesses [28]. The main drawbacks of machine learning methods are the lack of initial expert knowledge and the cold start challenge. On the other hand, the specification-based methods' main drawback is the lack of adaptation strategy in dynamic environments. Moreover, designing an accurate specification-based model for a single environment takes a lot of time and effort.

This work differs from the previous approaches as it eliminates each method's weakness by utilizing the other method advantages. We propose a novel approach to merge human knowledge and machine learning while keeping the process traceable, interpretable and clear to the smart environment's user and overcome the detection power of both main cores. This is an essential step toward a more reliable and adaptable situation awareness for smart environments. It provides the user with acceptable initial decisions based on a more general expert knowledge and customizes the experience to the user's expectations in each specific environment via an automatic process of requesting feedback. This will lighten the user's burden to customize the general framework and introduces an easy to setup environment that is still functional on the journey and does not fail on new and not experienced scenarios.

This paper introduces a new hybrid approach toward situation identification consistent with the architecture presented in [30], which is a specification-based approach that relies on situation templates. We show that this new hybrid approach is superior in accuracy, flexibility and adaptability compared with the prior work. To summarize, our main contributions are as follows:

- proposing an algorithm to utilize both situations templates as expert systems and decision trees as a learning approach in a unified system,
- creating a synthetic dataset and annotating it manually to analyze the approach,
- proposing an approach for extending an expert system and providing it with flexibility and adaptability using machine learning,
- achieving better results on the new dataset compared with both the expert system and the machine learning approach.

In the remainder of this paper, we review previous related works in Section 3. In Section 2, we discuss the challenges and the issues of possible modifications to the situation identification process. The method is presented in Section 4 and eventually the evaluation of the proposed method is described in Section 5.

2. BRIEF OVERVIEW OF CHALLENGES

In order to accurately detect situations in a smart environment, it is essential to understand the environmental characteristics. These characteristics are our motivation to develop a hybrid learning method rather than a specification-based approach.

The dynamic nature of active environments is already discussed in Section 1. It is our first key to design a self-adaptive method toward situation recognition.

Data corruption in smart environments is another challenge. Data corruption can be due to either broken sensors or uncertainty in sensor measurements. Uncertainty means that the data received from a sensor should not be considered 100% true [31]. Hence, the interpretation processes of these data should be able to handle errors and prevent incorrect reasoning on these uncertain inputs. Additionally, broken sensors result in either a shortage of required data input for interpretation or misleading input values.

Another challenge is the massive amount of data flowing from sensors to central processor units. Sensors are frequently producing values. These data are transformed and compiled into context information when received by a central processor unit. The processor unit is not aware of the usefulness of incoming data before analyzing them. Processing these data requires a significant amount of processing overhead [32]. If a

smart system has to store every image of environment sensors' data or process the whole input data at every data arrival, it would need a lot of resources and a significant amount of time to extract any meaningful information or retrieve an old stored data from the warehouses. Besides, the necessity of keeping historical information to recognize changes or patterns in an environment indicates the urgency of storing data into warehouses, which have to be considered in interpreting the new coming data. It justifies the big data challenge of smart environments without considering any appropriate filtering mechanism for data processing.

In the end, interaction with humans is the most challenging part of every HCI system. In a smart environment, if all tasks are done automatically or the user does not play any role in the decision cycle, the system is not fit for proper interaction with the user. The user must feel that the whole system is under his/her control and its mission is to help him/her to do his/her daily tasks easier and quicker. This type of interaction requires two primary considerations: first, the user must be aware of the system procedures and be able to change its behavior; second, the system must adapt itself to the user needs and facilitate the completion of routine or daily tasks.

Here, we provide some intuition for solving these challenges in the proposed model. First, to consider the dynamic nature of environments, we use a dynamic knowledge adaption through machine learning techniques. These techniques help modify the situation models whenever needed. Second, we handle uncertainty in two steps. First, the accuracy of sensor data is modeled as a relation between a sensor node and a confidence node in the context model mentioned in Fig. 1. Furthermore, machine learning techniques are used to detect broken sensors and eliminate them from the situation models. Third, as situation templates are used in the identification process, filtering is automatically applied as situation templates only contain predefined data types to check instead of checking all the inputs provided to them. The filtering mechanism decreases the overhead of processing all incoming data for interpreting the situation in real-time interaction while keeping all the data for offline processing.

In the end, we tackle the user interaction challenge by providing the ability to manipulate and adapt the situations. Our approach personalizes occurring situations by learning from user feedback, which relaxes the need for manual changes in situation templates. Hence, the user does not have to deal with the complexities of the situation templates' structure. Moreover, the changes are more reliable as modifications are based on frequent patterns.

3. RELATED WORK

There are three main categories of approaches toward situation identification.

Machine learning techniques. This type of methods considers situations as labels and sensors' data as inputs [33].

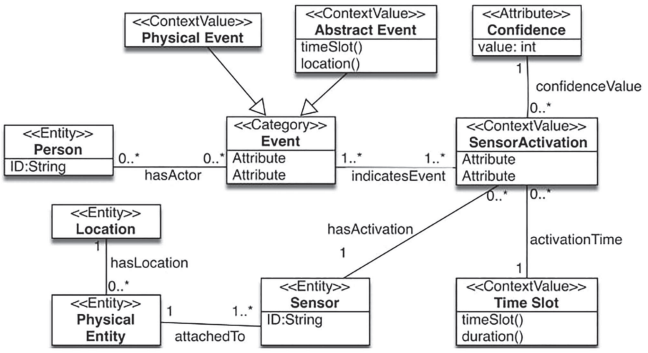


FIGURE 1. Core ontology for context modeling.

Neural network [20], naive Bayes [21], decision tree [22–24], hidden Markov models [23, 34], support vector machine [25], Bayesian networks [21, 28] and genetic algorithms [33] are types of techniques proposed on situation recognition.

Specification-based techniques. These methods rely on logical reasoning and rule engines [28, 35]. The first subcategory of such approaches is logic programming [36]. Utilizing logic programming for situation awareness is by specifying each situation as a set of rules and logical operations. The source of limitations in these approaches is the limited expressiveness in description logic constraints.

Another subcategory is the ontology modeling [2, 18, 26, 27, 35, 37]. An ontology is defined as an explicit specification for a concept [38]. It defines each concept by its definition and relations to other concepts. This technique relies on designing an ontology, which represents the problem domain. The main benefit of using ontology is that the modeled knowledge is shareable, transferable and understandable by both machines and humans [39]. Matheus *et al.* [2] proposed a core ontology for situation-aware applications. Their approach is based on the claim that the knowledge about objects' relationships is the core requirement for situation identification. This core ontology should be customized according to each specific situation. In the proposed ontology, situations are related to goals, rules and physical objects. Pearson *et al.* [27] developed two separate ontology models for context modeling and situation modeling. The former is presented in Fig.1. This ontology describes an event with the use of the sensors' value, the location of the sensors, the timestamp of an event and the reliability of sensors' metric.

Modeling by situation templates is another subcategory of specification-based approaches. Situation templates are one-directional XML trees with a root labeled as the situation and written in XML [30, 40–43].

Spatial and temporal logics are used to model reasoning over the location and the time of events [44]. Besides, fuzzy logic is widely used to control the uncertainty of sensor values [45, 46].

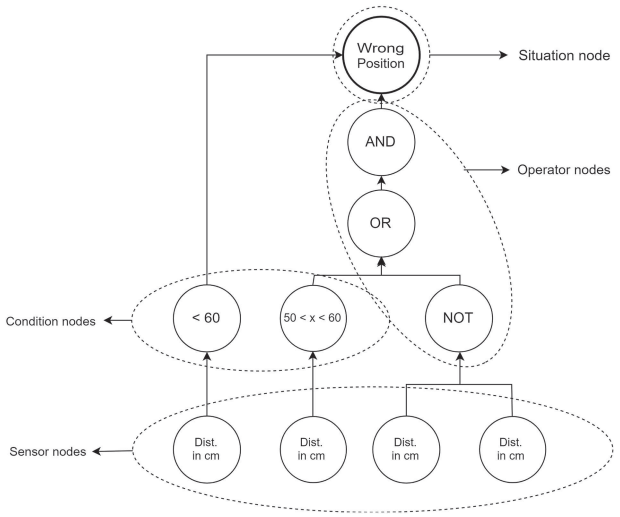


FIGURE 2. A situation template sample.

Hybrid approaches are the recent point of view for many data processing communities [47–49] to benefit from multiple training techniques or knowledge-aware resources together. In this context, hybrid learning is used to indicate the combination of specific-based methods and machine learning methods. Yuan and Herbert [50] introduced a hybrid learning algorithm in healthcare scope as a context-aware real-time assistant using rule-based and case-based reasonings. Using machine learning techniques to generate dynamic association rules and adding the sound rules to adapt ontology is another investigated method [29].

Sukor *et al.* [51] has argued that using these hybrid approaches helps in adapting to personalized situations and is more compatible with human behavior. They proposed a hybrid approach with a knowledge-aware starting point and extended their method using SVM as the training technique to adapt the knowledge to the user's varying actions. Although their approach is similar to ours, our usage of situation templates yields an interpretable decision process, which is more suitable for smart environments.

In this paper, we present a hybrid learning algorithm based on situation templates and decision tree combining as online and offline reasoning tools. Because of the specific structure of decision trees, they are easily integrated with situation templates. Additionally, unlike other machine learning techniques that do not specify the logic of producing the final label, the label generation process of decision trees is human-understandable.

4. THE PROPOSED HYBRID APPROACH

We propose a model to solve the challenges mentioned in Section 2. Our model is built on top of the SitOPT framework [30]. SitOPT is a three-layer framework containing a sensor layer, a situation recognition layer and a workflow layer. The sensor layer handles communication regarding physical objects and hardware. It gathers sensor data and transforms them into context information. The situation recognition layer identifies the situations according to the received input from the sensor layer. Finally, the identified situation is transferred to the workflow layer, where it is translated into a set of executable tasks that affect the environment through appropriate actuators. SitOPT is incapable of capturing the dynamic nature of environments because it is a specification-based algorithm. The SitOPT [30] deployment is suggested to be done with the Node-RED platform, which executes data flows in the clouds, local and any other devices [30].

Our modifications improve the accuracy and adaptability of SitOPT's situation recognition layer in dynamic environments. Our approach utilizes the power of observed data to personalize an initial expert system and is able to deal with unseen situations. SitOPT's [30] situation templates have a tree structure in which nodes have one of four types: operator, condition, sensor and situation (Fig. 2). Templates are constructed by expert users. Thus, errors may be due to the lack of environment understanding by the experts, no consideration of sensor corruption and no consideration of the environment's living agent behavior. In addition, events such as adding a new sensor, removing a sensor or a change in human behavior are other factors that change the environment specifications. Thus, the need for an adaptation strategy is evident.

Figure 3 provides an overview of the proposed model. The machine learning unit captures sensors' images[2] and user's feedback to generate a decision tree. The situation repository is the storage for the initial and updated situation models. The enhancer unit merges the decision tree, made by the machine learning unit, with the situation templates. The recognizer unit detects the ongoing situation by checking situation models in the repository against the incoming data. Finally, the workflow fragment repository maps the selected situation to different environmental functions. The workflow fragment repository, the situation repository and the recognizer unit of this architecture are adopted from SitOPT [30].

The machine learning unit uses the C4.5 algorithm [52] to build a decision tree [53], which helps to adapt the situation models to a specific environment. Figure 4 shows an example of a decision tree. Decision trees and situation templates have a similar structure, but their data processing direction are different from each other. In a decision tree, labels are determined by traversing the tree from the root to the leaf nodes, while in a situation template, inputs are cross-checked with all possible paths from the leaves to the root. Moreover, a decision tree can

[2] A sensor image is a snapshot of one particular moment along with all the sensor values and the decision made by the system or the user.

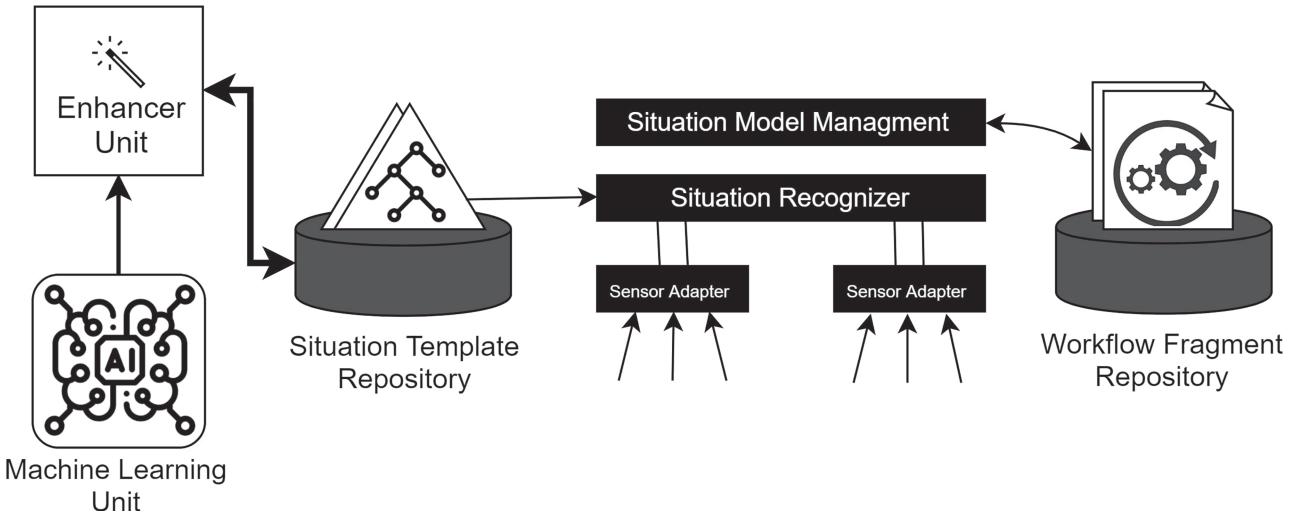


FIGURE 3. The overall view of the architecture.

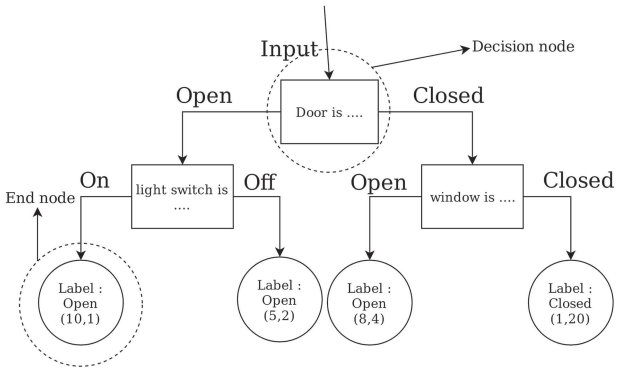


FIGURE 4. A decision tree sample.

distinguish between several classes, while a situation template is a boolean classifier.

Almost all tasks in smart environments are semi-automated or launched with an alert to the user. The user's responses to the alerts and approvals of the recommended actions made by the semi-automated process provide the system with plenty of labeled samples. The machine learning algorithm uses these labeled data to train the aforementioned decision tree. In this decision tree, each label node is associated with a precision[3] value and a total number of the occurrences of the relevant label in the training examples.

The enhancer unit follows the steps below to adapt the situation templates by considering the collected observations.

1. Convert the decision tree and situation templates to DNF[4] trees.
2. Update the DNF trees from situation templates based on the decision tree.
3. Convert the updated DNF trees to situation templates.
4. Store the updated situation templates back to the situation template repository.

The enhancer unit procedure is shown in Fig. 5. After the first step, each label (situation) is described by two DNF trees: a tree from the decision tree paths and a second tree from the relevant situation template in the situation repository.

As shown in Fig. 6, each generated DNF tree consists of condition nodes, 'AND' operator nodes, an 'OR' operator node and a situation node. A path in a DNF tree is a sub-tree that starts with an 'AND' operator node. Each path shows a required set of conditions that indicates the occurrence of the situation located at the root. As each branch of the decision tree results in exactly one path in a DNF tree, this path would inherit the precision value and the number of total positively detected examples.

Updating the situation templates is a three-step procedure. First, Algorithm 1 generates a set of similar pairs of paths. PT_i^s represents the path i in the DNF resulted from the situation template for the situation s and PD_j^s represents the path j in the DNF resulted from the decision tree for the situation s. Also, SM denotes a set of similar pairs where

$$SM = \{(PT_i^{s_1}, PD_j^{s_1}), (PT_a^{s_1}, PD_b^{s_1}), ..., (PT_l^{s_n}, PD_k^{s_n})\}.$$

[3] TP/(TP+FP)

[4] Disjunctive normal form

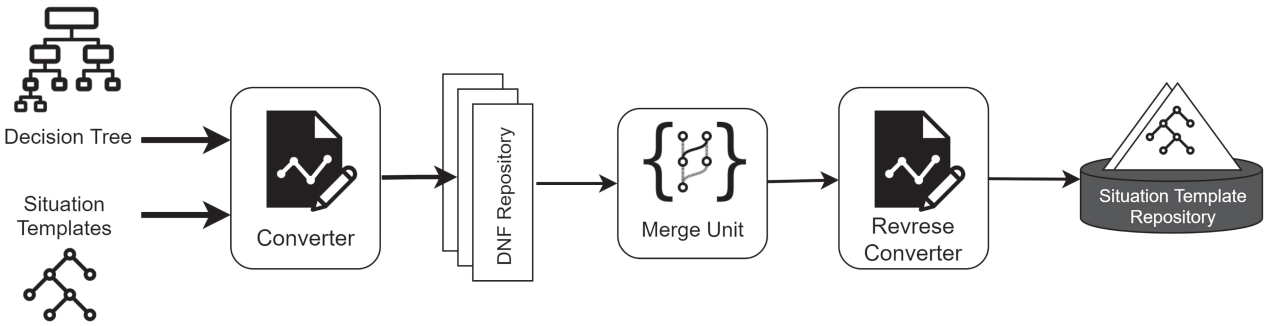


FIGURE 5. The enhancer process overview.

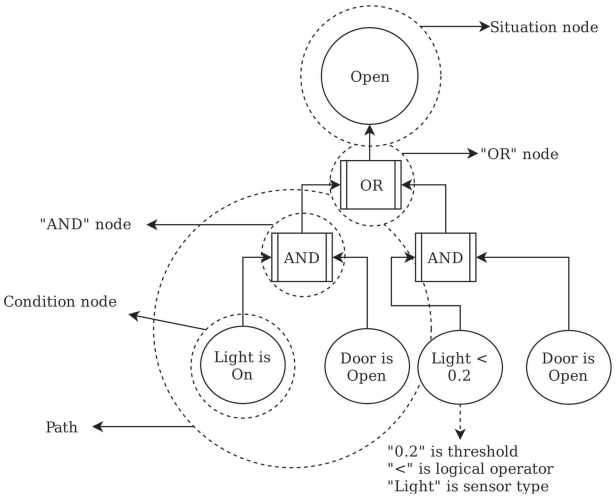


FIGURE 6. DNF tree.

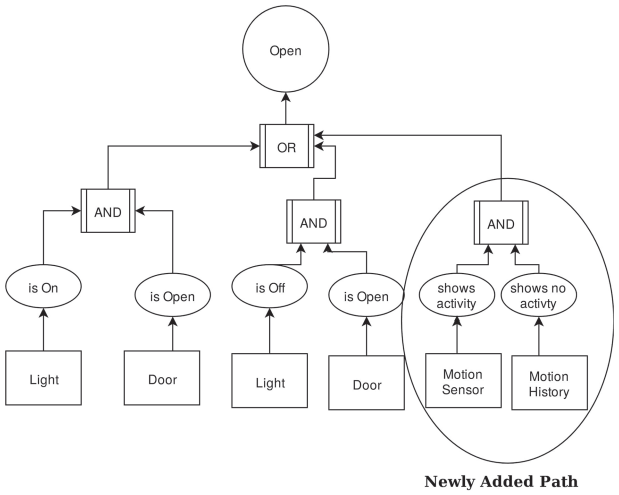


FIGURE 7. Adding new paths to situation template.

Each pair represents a similar set of requirements to detect the same situation and is attached with a similarity score, which shows the probability of the alignment. The similarity score of a pair is computed based on its paths' condition nodes and the value of their defined thresholds. Condition nodes can be similar if and only if they have the identical sensor type and the logical operator.

Second, the pairs are pruned by discarding all pairs whose similarity score is below a threshold. the default threshold in the experiments is 0.6. Finally, in each selected pair $(PT_l^{s_i}, PD_k^{s_i})$, $PT_l^{s_i}$ is updated based on $PD_k^{s_i}$. In order to apply the update process on a pair, we define two boolean variables of paths' and labels' reliability on the decision tree DNFs. These values are used to ensure lower risk on updating the situation templates. Formula (1) shows the computation of path reliability. Furthermore, we define label confidence and label cardinality. Label confidence is the minimum precision value of the paths in the DNF trees of a specific situation, and the label cardinality shows the total number of the available training data for that situation. Using Formulas (3) and (4), label reliability is computed based on Formula (5).

$$Reliability(path, P^5, C^6) = \begin{cases} 0 & : P < 0.65 \vee C < 10 \\ 1 & : o.w. \end{cases} \quad (1)$$

$$train - data = \{(x, y) : x = input, y = label\} \quad (2)$$

$$Card(A) = \mid \{(a, b) \mid (a, b) \in train - data \wedge b = \text{"A"}\} \mid \quad (3)$$

$$Conf(A) = min\{x_{precision} : x_{precision} \in A(precision)\} \quad (4)$$

5 Precision
6 Total number of training data with this label

TABLE 1. The types of sensors.

Sensors	Output	Description	Wrong outputs on
Motion sensor	{0,1}	Detects human motion	Not moving humans
Boolean light	{0,1}	Detects the light status	
Fuzzy light	[0,1]	Detects the light status	
Noise sensor	[0,1]	Detects room's noise	Outside noises
TV sensor	{0,1}	Detects the TV status	

$$Label - Reliability(A) = \begin{cases} 0 : Conf(A) < 0.8 \vee Card(A) < 100 \\ 1 : o.w. \end{cases} \quad (5)$$

At the final step, the following updates on the pairs containing a reliable path in the decision tree are possible:

1. adding a new path to the situation template,
2. removing a path from the situation template,
3. updating a threshold value in a path from the situation template.

When a reliable path for situation *s* from the decision tree is not present in any pair of the selected set, a new path is added to the situation template's DNF tree of situation *s*. Figure 7 shows an example of adding a new path to a situation template.

Algorithm 1: Making the similarity set

```
Input: dtdnf (a labeled decision tree DNF),
       stdnf (a labeled situation template DNF)
Output: The array of similarities between paths
stpArray ← stdnf paths
dtpArray ← dtdnf paths
similarArray ← [ ]
foreach dtp in dtpArray do
    max = 0
    match = null
    foreach stp in stpArray do
        similarity = similarity − computer(stp, dtp) //which is describe in Algorithm 2
        if similarity >= 0.6 and similarity > max then
            max = similarity
            match = stp
        end
    end
    remove match from stpArray
    add [match, dtp, max] to similarArray
end
return similarArray
```

Algorithm 2: Similarity Computing Algorithm

```
Input: dtp (a path in decision tree DNF), stp (a path in situation template DNF)
Output: a similarity score between 0 and 1
stpLeafs ← stp leaves of the situation template DNF
dtpLeafs ← dtp leaves of the decision tree DNF
count ← 0 length ← max(stpLeafs.length, dtpLeafs.length)
foreach snode in stpLeafs do
    foreach dnode in dtpLeafs do
        if snode contains Threshold and dnode contains Threshold and snode comprator = dnode comprator then
            if snode.SensorType = dnode.SensorType && [snode threshold - dnode threshold] <= 0.25 then
                count = count + 1; break;
            end
        end
        else if snode doesn't contain Threshold and dnode doesn't contain Threshold
            if snode.SensorType = dnode.SensorType && snode.Value = dnode.Value then
                count = count + 1; break;
            end
        end
    end
end
return count ÷ length
```

If the label for situation *s* is reliable in the decision tree, the paths from the situation template for situation *s* that do not have a representative pair in the selected set *SM* are removed. Removing these paths decreases the execution latency and increases the accuracy of the situation recognition process. The accuracy is improved due to the reduction of erroneous recognition and fewer misinterpretation of received data. Moreover, this process helps to remove paths that contain broken sensors. However, as the decision tree is built on top of observations, there are rare paths considered by the initial templates but not reflected in the decision tree. Hence, there is no match for such conditions in the selected pairs. In order to prevent the removal of such paths, we allow the initial templates to mark these paths with a flag. The flagged paths are never removed, whether they appear in the similarity set or not.

TABLE 2. List of sensors in each room.

Rooms/ sensors	Motion sensor	Boolean light	Fuzzy light	Noise sensor	Television sensor
Working room	1	0	1	1	1
Management room	1	0	1	0	0
Rest room	1	1	0	0	0

TABLE 3. The defined situations in the environment.

Sensors	Description
Opening	The company is opened (it was previously closed).
Closing	The company is closed (It was previously open).
Working	The manager or the staff are working.
Educating	An educating session is active in company.

When the initial templates are designed carefully, the threshold update happens more frequently than previous types of modifications. Even if the experts do well on modeling, determining the thresholds of condition nodes is barely possible without analyzing the specific environment for a long period. For example, some may read books in dimmer light than others or some may prefer a different temperature threshold for the air conditioning system to start cooling. Therefore, thresholds are not accurately predictable before running the system for a while and are not evident at the initial step of the situation modeling. Thus, this type of update personalizes the templates for a specific environment.

In the threshold update process, the thresholds of the condition nodes of a path from a situation template DNF tree are updated according to the condition nodes in the other path in the pair. For instance, this process can change the conditional comparison $LIGHT - SENSOR - 1 > 0.4$ to $LIGHT - SENSOR - 1 > 0.8$.

The updated DNF trees of situation templates are transformed back to their original format and replace the existing templates in the repository. If users change their behavior or some new sensors are added to the system, the machine learning data have to be removed and started again. As machine learning algorithms are built on top of the data provided to them, they usually reflect the long-term conditions and do not reflect new changes at a good pace. Therefore, the user should be able to redefine the training data.

5. EXPERIMENT

5.1. Dataset

We provide a synthetic dataset containing environmental observations in a small company. A single company consisted of three different rooms is simulated. The types of sensors in each room are listed in Table 1. The smart environment caches the values of sensors every 1 second to enable the detection of changes. The existence of sensors in each room is available in Table 2.

Based on these sensors, five situations are defined. These situations are listed in Table 3.

We generate random data from the sensors. These data are annotated manually and split into training and test sets. The generated data are double-checked to be feasible sensors' images at two points. First, the initial images are tested by a rule engine to verify the possibility of occurrence. Secondly, the labeled data are checked by some simple rules to avoid human mistakes. However, we have left 2% of the errors untouched so that it can reflect the erroneous user feedback in real-world scenarios. The manual labeling process has been done by more than 10 different people to reflect different opinions and change of behavior as we feed the data increasingly to the system.

Furthermore, the generated data are splitted into seven parts. One part is considered as test data and is not provided to the machine learning unit. Other parts represent a limited working period. Each part involves 200–250 sensors' images. This consideration is mandatory because it is unrealistic to assume that all sensors' images are available at the initial step. Thus, we simulate the time by running our proposed model in six steps while adding one part to the training data at each step, assuming they were generated increasingly through time.

5.2. Evaluation

As experts and their defined situations' models play an essential role in the outcome of our approach, we evaluate the proposed method by defining two different sets of initial templates. The first set contains good starting models that are highly accurate, and the second set is poorly designed and is used to test the dependency to the starting knowledge and the pace of improvement.

Finally, for each time step, we use the test data to evaluate the initial situation templates, the updated situation templates and the decision tree performance. The result of evaluation on bad starting models is plotted in Fig. 8 and the good start evaluation is depicted in Fig. 9. In all figures, the solid squared line represents the initial situations templates, the dashed line with stars represents the updated templates and the dashed line with cross marks represents the decision tree. In each row of the figure, the left plot shows the accuracy comparison, the central one represents the precision comparison and the right one shows the evaluation on recall. For a brief overview, the results of the accuracy of bad start initialization are listed in Table 4.

The results show that our approach's accuracy, precision and recall are either better or equal to the initial situation templates. However, our approach is superior in many cases. Moreover, results show that updates do not happen based on an inaccurate decision tree. An essential factor for evaluation is the reliability

TABLE 4. The comparison with baselines on bad initial starts on accuracy of predictions.

Step 1	Step 2				Step 4				Step 6			
	W	O	C	E	W	O	C	E	W	O	C	E
DT	42.55	**100**	84.2	76.67	87.23	**100**	**100**	83.33	91.48	**100**	**100**	83.33	93.617	**100**	**100**	83.33
ST	**83**	91.5	96.5	91	83	91.5	96.5	91	83	91.5	96.5	91	83	91.5	96.5	91
HL	**83**	98.5	**98**	**99**	**97**	96	**100**	**99**	**99.5**	**100**	**100**	**99**	**100**	**100**	**100**	**99**

W indicates working; O, opening; C, closing; E, educating; HL, hybrid learning; ST, situation template; DT, decision tree.

thresholds. Hence, it introduces a new hyper-parameter to the network.

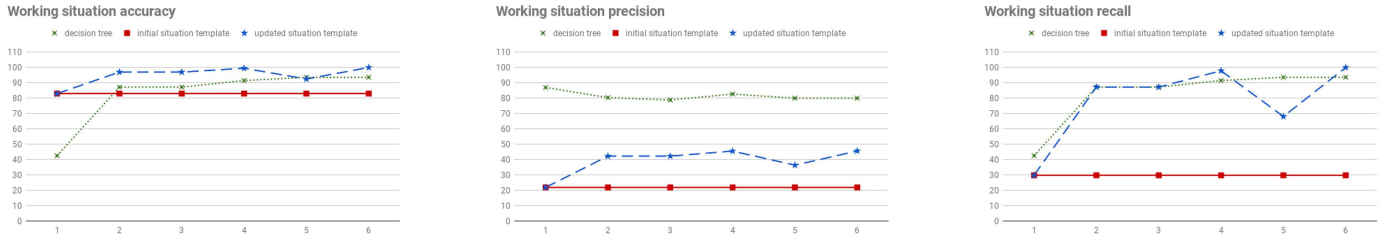
Using a data-driven approach increases the flexibility and adaptability of the models in comparison with logical models. Furthermore, using machine learning in combination with logical approaches results in more adjustable models. Moreover, our approach's adaptability is due to its personalized prediction model for each environment via numerous updates. Updating the thresholds inside comparison nodes is the main reason to acquire more adaptability in models. Finally, a static situation model is not able to reflect changes regarding adding or removing a sensor. On the contrary, our model is more flexible to the environment's changes due to capturing those modifications and adapting the situation models accordingly.

6. LIMITS AND FUTURE WORK

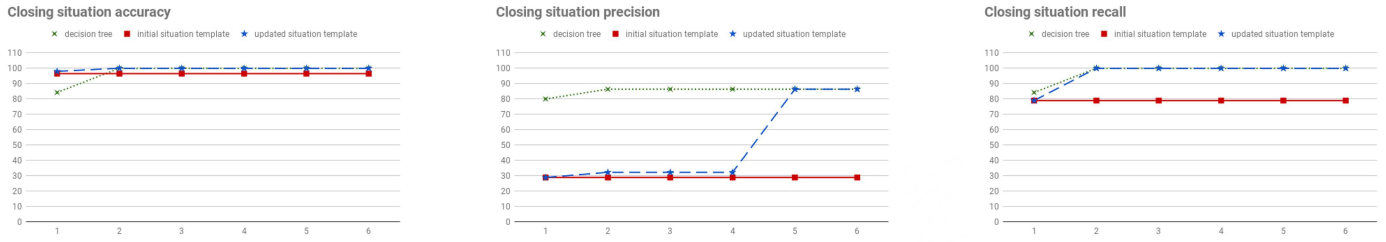
As discussed in Section 1, the importance of situation awareness in smart environments is undeniable. However, there are still a lot of challenges unresolved by prior work, given real-world scenarios. Despite all the benefits of deep neural networks and their successful implementations, there is still a need for more sophisticated approaches to utilize human knowledge alongside such structures. However, deep neural networks' black-box architecture is the main drawback preventing integrating prior knowledge in their learning architecture design. Under such considerations, it is an excellent opportunity to look back at traditional machine learning approaches and investigate the possibility of logical integration with those algorithms. This work proposes an approach that uses a decision tree as a traditional machine learning approach and integrates a logical representation of human knowledge in the form of trees to augment the decision tree's output. Here, we have shown an example of possible integration of logic and traditional machine learning approaches to encourage the community to look back at previous achievements and probe for opportunities to expand further.

As our research is limited because of no access to a functioning smart environment, we have evaluated our method with a simulated scenario. However, to compare the proposed approach's performance on more complicated scenarios, it is mandatory to evaluate with a real-world functioning smart environment because human behavior is more unexpectable than considered in any simulation process. We have implemented our approach based on Python programming language and decision tree implemented in Scikit-learn [54] package. The implementation code is publicly available on GitHub[7].

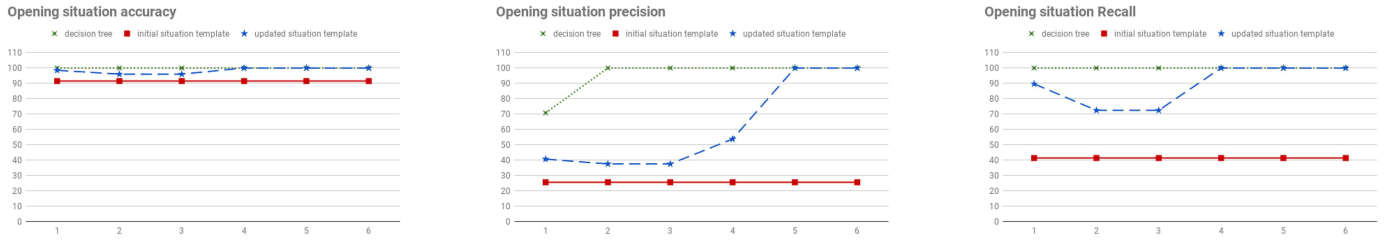
[7] https://github.com/hfaghihi15/SITRTS



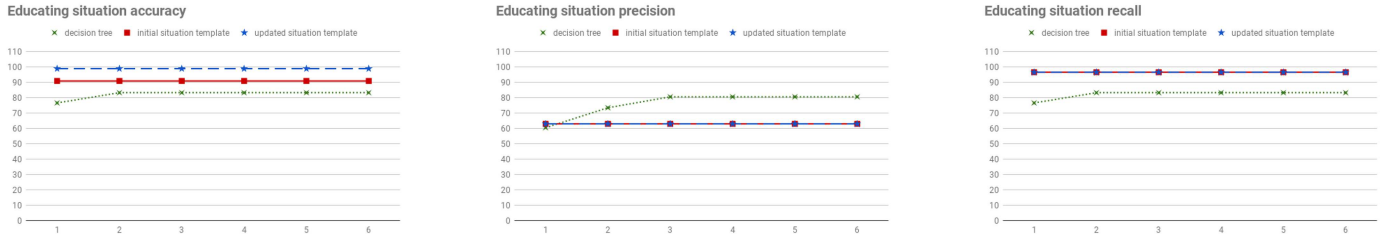
(a) Working situation evaluation



(b) Closing situation evaluation



(c) Opening situation evaluation



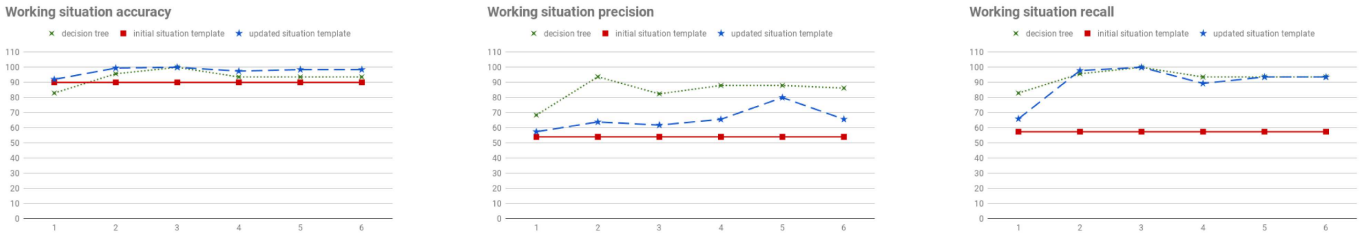
(d) Educating situation evaluation

FIGURE 8. Bad start evaluation (accuracy, precision, recall).

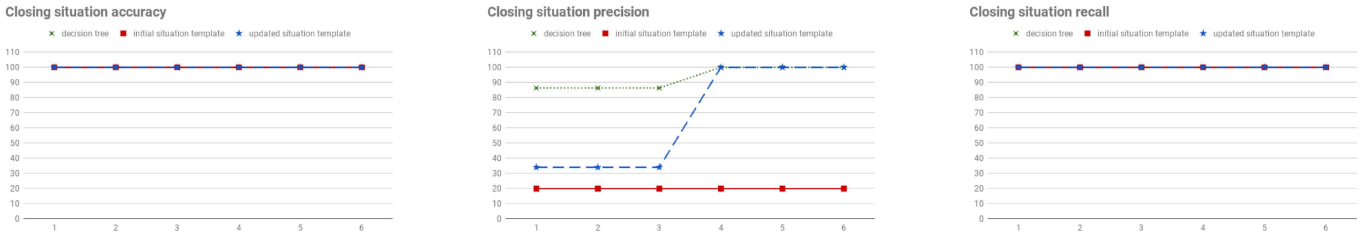
One possible future work is investigating the possibility of merging knowledge graphs and deep neural networks, enabling the usage of more powerful machine learning computation integrated with a more sophisticated human knowledge source. Furthermore, processing a large amount of data requires significant computational resources, which may not be available for all experiments. Thus, considering the growing number of sensors available in smart environments and the interpretation complexity for processing them appropriately, there is a need for a more intelligent filtering method. Accordingly, another possible future direction is designing a filtering approach based on predicting the situation on a subset of observed sensor data and how to find the optimal subset.

7. CONCLUSION

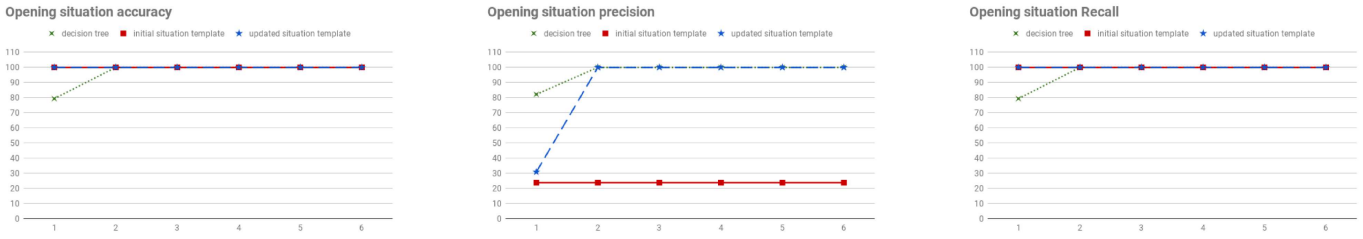
In this paper, we proposed a hybrid learning method for situation identification, which utilizes situation templates and



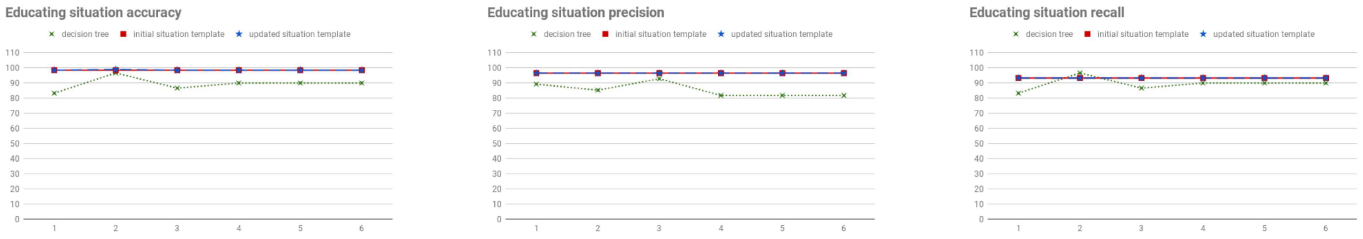
(a) Working situation evaluation



(b) Closing situation evaluation



(c) Opening situation evaluation



(d) Educating situation evaluation

FIGURE 9. Good start evaluation (accuracy, precision, recall).

a decision tree. This method is evaluated against a growing collection of data to simulate time in the real-world implementations of a smart environment. The results showed that our approach guarantees better accuracy in every step of integration compared with the initial situation templates. This work can be extended by considering a proper feedback gathering mechanism that does not distract the user frequently. Another future directing is the automatic extraction of new events and situations by clustering the observations of the user's sequence of actions.

8. DATA AVAILABILITY STATEMENT

The authors confirm that the data supporting the findings of this study are available within the article and/or its supplementary materials. The implementation code and data used for this research can be found on GitHub[8] .

[8] https://github.com/hfaghihi15/SITRTS

REFERENCES

[1] Mukhopadhyay, S.C. and Suryadevara, N.K. (2014) *Internet of Things: Challenges and Opportunities*. Internet of Things, pp. 1–17. Springer.

[2] Matheus, C.J., Kokar, M.M. and Baclawski, K. (2003) A Core Ontology for Situation Awareness. In *Proc. Sixth Int. Conf. Information Fusion*, pp. 545–552.

[3] D'Aniello, G., Loia, V. and Orciuoli, F. (2015) A multi-agent fuzzy consensus model in a Situation Awareness framework. *Appl. Soft Comput.*, 30, 430–440.

[4] Dey, A.K. (2001) Understanding and using context. *Pers. Ubiquit. Comput.*, 5, 4–7.

[5] Abowd, G.D., Dey, A.K., Brown, P.J. and Davies, E.A. (1999) Towards A Better Understanding of Context and Context-Awareness. In *Int. Symposium on Handheld and Ubiquitous Computing*, pp. 304–307. Springer.

[6] Al-Sultan, S., Al-Bayatti, A.H. and Zedan, H. (2013) Context-Aware driver behavior detection system in intelligent transportation systems. *IEEE Trans. Veh. Technol.*, 62, 4264–4275.

[7] Wan, J., Zhang, D., Zhao, S., Yang, L.T., and Lloret, J. (2014) Context-Aware vehicular cyber-physical systems with cloud support: Architecture, challenges, and solutions. *IEEE Commun. Mag.*, 52, 106–113.

[8] Santa, J. and Gomez-Skarmeta, A.F. (2009) Sharing context-aware road and safety information. *IEEE Pervasive Comput.*, 8, 58–65.

[9] Ferreira, J.C., Silva, H., Afonso, J.A. and Afonso, J.L. (2018) Context aware advisor for public transportation. *IAENG Int. J. Comput. Sci.*, 45, 74–81.

[10] Rehena, Z. and Janssen, M. (2018) Towards a Framework for Context-Aware Intelligent Traffic Management System in Smart Cities. In *Companion of the The Web Conf. 2018 on The Web Conf. 2018*, pp. 893–898. International World Wide Web Conferences Steering Committee.

[11] Al-Turjman, F. (2018) QoS-Aware data delivery framework for safety-inspired multimedia in integrated vehicular-IoT. *Comput. Commun.*, 121, 33–43.

[12] Al-Turjman, F., Ever, E. and Zahmatkesh, H. (2018) Small Cells in the Forthcoming 5G/IoT: Traffic Modeling and Deployment Overview. In *Smart Things and Femtocells*, pp. 17–82. CRC Press.

[13] Urbieta, A., González-Beltrán, A., Mokhtar, S.B., Hossain, M.A. and Capra, L. (2017) Adaptive and context-aware service composition for IoT-based smart cities. *Futur. Gener. Comput. Syst.*, 76, 262–274.

[14] Abdellatif, A.A., Mohamed, A., Chiasserini, C.F., Tlili, M. and Erbad, A. (2019) Edge computing for smart health: Context-Aware approaches, opportunities, and challenges. *IEEE Netw.*, 33, 196–203.

[15] Aborokbah, M.M., Al-Mutairi, S., Sangaiah, A.K. and Samuel, O.W. (2018) Adaptive context aware decision computing paradigm for intensive health care delivery in smart cities—A case analysis. *Sustain. Cities Soc.*, 41, 919–924.

[16] Casino, F., Patsakis, C., Batista, E., Postolache, O., Martínez-Ballesté, A. and Solanas, A. (2018) Smart Healthcare in the IoT Era: A Context-Aware Recommendation Example. In *2018 Int. Symposium in Sensing and Instrumentation in IoT Era (ISSI)*, pp. 1–4. IEEE.

[17] Al-Turjman, F. and Alturjman, S. (2018) Context-sensitive access in industrial internet of things (IIoT) healthcare applications. *IEEE Trans. Ind. Inf.*, 14, 2736–2744.

[18] Kolbe, N., Zaslavsky, A., Kubler, S., Robert, J. and Le Traon, Y. (2017) Enriching a Situation Awareness Framework for IoT with Knowledge Base and Reasoning Components. In *International and Interdisciplinary Conf. Modeling and Using Context*, pp. 41–54. Springer.

[19] Endsley, M.R. (1995) Measurement of situation awareness in dynamic systems. *Hum. Factors*, 37, 65–84.

[20] Yang, J.-Y., Wang, J.-S. and Chen, Y.-P. (2008) Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers. *Pattern Recogn. Lett.*, 29, 2213–2220.

[21] Korpipää, P., Koskinen, M., Peltola, J., Mäkelä, S.M. and Seppänen, T. (2003) Bayesian approach to sensor-based context awareness. *Pers. Ubiquit. Comput.*, 7, 113–124.

[22] Alkhomsan, M.N., Hossain, M.A., Rahman, S.M.M. and Masud, M. (2017) Situation awareness in ambient assisted living for smart healthcare. *IEEE Access*, 5, 20716–20725.

[23] Lee, S.-Y. and Lin, F.J. (2016) Situation Awareness in a Smart Home Environment. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pp. 678–683. IEEE.

[24] Logan, B., Healey, J., Philipose, M., Tapia, E.M. and Intille, S. (2007) A Long-Term Evaluation of Sensing Modalities for Activity Recognition. In *Int. Conf. Ubiquitous Computing*, pp. 483–500. Springer.

[25] Kanda, T., Glas, D.F., Shiomi, M., Ishiguro, H. and Hagita, N. (2008) Who will be the Customer? A Social Robot that Anticipates People's Behavior from their Trajectories. In *Proc. 10th Int. Conf. Ubiquitous Computing*, pp. 380–389. ACM.

[26] Ghimire, S., Luis-Ferreira, F., Nodehi, T. and Jardim-Goncalves, R. (2017) IoT based situational awareness framework for real-time project management. *Int. J. Comput. Integr. Manuf.*, 30, 74–83.

[27] Pearson, R., Donnelly, M.P., Liu, J. and Galway, L. (2016) Generic Application Driven Situation Awareness via Ontological Situation Recognition. In *2016 IEEE International Multi-Disciplinary Conf. Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, pp. 131–137. IEEE.

[28] Ye, J., Dobson, S. and McKeever, S. (2012) Situation identification techniques in pervasive computing: A review. *Pervasive Mob. Comput.*, 8, 36–66.

[29] Kim, M., Kang, H., Kwon, S., Lee, Y., Kim, K. and Pyo, C.S. (2017) Augmented Ontology by Handshaking with Machine Learning. In *2017 19th Int. Conf. Advanced Communication Technology (ICACT)*, pp. 740–743. IEEE.

[30] Hirmer, P., Wieland, M., Schwarz, H., Mitschang, B., Breitenbücher, U., Sáez, S.G. and Leymann, F. (2017) Situation recognition and handling based on executing situation templates and situation-aware workflows. *Computing*, 99, 163–181.

[31] Alberti, A.M. and Singh, D. (2013) Internet of Things: Perspectives, Challenges and Opportunities. In *International Workshop on Telecommunications (IWT 2013)*, pp. 1–6.

[32] Sukode, S., Gite, S. and Agrawal, H. (2015) Context aware framework in IoT: A survey. *Int. J. Adv. Sci. Technol.*, 4. http://warse.org/pdfs/2015/ijatcse01412015.pdf.

[33] Yang, K., Wang, J., Bao, L., Ding, M., Wang, J. and Wang, Y. (2016) Towards Future Situation-Awareness: A Conceptual Middleware Framework for Opportunistic Situation Identification. In *Proc. 12th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pp. 95–101. ACM.

[34] Damarla, T. (2008) Hidden Markov Model as a Framework for Situational Awareness. In *2008 11th Int. Conf. Information Fusion*, pp. 1–7. IEEE.

[35] Lakehal, A., Alti, A. and Roose, P. (2019) A semantic event based framework for complex situations modeling and identification in smart environments. *Int. J. Adv. Comput. Res.*, 9, 212–221.

[36] Barwise, J. (1986) The Situation in Logic—I. In *Studies in Logic and the Foundations of Mathematics*, pp. 183–203. Elsevier.

[37] Machado, A., Maran, V., Augustin, I., Wives, L.K. and de Oliveira, J.P.M. (2017) Reactive, proactive, and extensible situation-awareness in ambient assisted living. *Expert Syst. Appl.*, 76, 21–35.

[38] Gruber, T.R. (1993) A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5, 199–220.

[39] Chen, L., Nugent, C., Mulvenna, M., Finlay, D. and Hong, X. (2009) Semantic Smart Homes: Towards Knowledge Rich Assisted Living Environments. In *Intelligent Patient Management*, pp. 279–296. Springer.

[40] Mormul, M., Hirmer, P., Wieland, M. and Mitschang, B. (2017) Situation model as interface between situation recognition and situation-aware applications. *Comput. Sci. Res. Dev.*, 32, 331–342.

[41] Hirmer, P., Wieland, M., Schwarz, H., Mitschang, B., Breitenbücher, U. and Leymann, F. (2015) SitRS—A Situation Recognition Service Based on Modeling and Executing Situation Templates. In *Proc. 9th Symposium and Summer School on Service-Oriented Computing*, pp. 113–127.

[42] da Silva, A.C.F., Hirmer, P., Wieland, M. and Mitschang, B. (2016) SitRS XT-towards near real time situation recognition. *J. Inform. Data Manag.*, 7, 4.

[43] Zweigle, O., Häussermann, K., Käppeler, U.-P. and Levi, P. (2009) Supervised Learning Algorithm for Automatic Adaption of Situation Templates Using Uncertain Data. In *Proc. 2nd Int. Conf. Interaction Sciences: Information Technology, Culture and Human*, pp. 197–200. ACM.

[44] Cook, D.J., Augusto, J.C. and Jakkula, V.R. (2009) Ambient intelligence: Technologies, applications, and opportunities. *Pervasive Mob. Comput.*, 5, 277–298.

[45] Ranganathan, A., Al-Muhtadi, J. and Campbell, R.H. (2004) Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Comput.*, 3, 62–70.

[46] Ke, M., Zhu, B., Zhao, J. and Deng, W. (2018) Automatic Drive Train Management System for 4WD Vehicle Based on Road Situation Identification. Technical Report. SAE Technical Paper.

[47] Pedersen, S.A., Yang, B. and Jensen, C.S. (2020) A Hybrid Learning Approach to Stochastic Routing. In *2020 IEEE 36th Int. Conf. Data Engineering (ICDE)*, pp. 1910–1913. IEEE.

[48] Pedersen, S.A., Yang, B. and Jensen, C.S. (2020) Anytime stochastic routing with hybrid learning. *Proc. VLDB Endow.*, 13, 1555–1567.

[49] Nouh, R.M., Lee, H.-H., Lee, W.-J. and Lee, J.-D. (2019) A smart recommender based on hybrid learning methods for personal well-being services. *Sensors*, 19, 431.

[50] Yuan, B. and Herbert, J. (2014) Context-aware hybrid reasoning framework for pervasive healthcare. *Pers. Ubiquit. Comput.*, 18, 865–881.

[51] Sukor, A.S.A., Zakaria, A., Rahim, N.A., Kamarudin, L.M., Setchi, R. and Nishizaki, H. (2019) A hybrid approach of knowledge-driven and data-driven reasoning for activity recognition in smart homes. *J. Intell. Fuzzy Syst.*, 36, 4177–4188.

[52] Quinlan, J.R. (1993) *C4. 5: Programs for Machine Learning*, vol. 38, p. 48. Morgan Kauffmann.

[53] Safavian, S.R. and Landgrebe, D. (1991) A survey of decision tree classifier methodology. *IEEE Trans. Syst. Man Cybern.*, 21, 660–674.

[54] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J. (2011) scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12, 2825–2830.