Journal of Ambient Intelligence and Smart Environments 0 (2010) 1
IOS Press

Exploring Semantics in Activity Recognition Using Context Lattices

Juan Ye * and Simon Dobson
School of Computer Science, University of St Andrews, St Andrews, Fife, UK, KY16 9SX.
E-mail: {ye, sd}@cs.st-andrews.ac.uk.

Abstract. Studying human activities has significant implication in human beneficial applications such as personal healthcare. This research has been facilitated by the development of sensor technologies in pervasive computing with a large quantity of observational data collected about environments and user actions. By mining these data, traditional machine learning techniques have made great progress in recognising activities, but due to the increasing number of sensors and complexity of activities, they are subject to feasibility and scalability. These techniques may benefit from the inclusion of semantic information about the nature and relationships of sensor data and activities being observed. We introduce a new data structure, the *context lattice*, which allows designers to capture and explore this sort of knowledge. We demonstrate how context lattices can be used to infer human activities with the inclusion of such knowledge. We present comprehensive evaluations of the system against two third-party smart-home data sets, and demonstrate that our approach compares favourably with traditional analytic techniques in many circumstances. We conclude with a discussion of the strengths and weaknesses of context lattices in activity recognition.

Keywords: activity recognition, smart home environments, semantics

1. Introduction

Human activity recognition has gained increased attention in recent years with the development of in-home pervasive computing technologies, including small and highly portable sensors that can be attached to the body and to everyday objects [1,2]. The deployment of these technologies has brought about new opportunities for studying human activities. Also the increase in the amount and variety of sensors fosters the richer semantics underlying them; for example, location data with varying levels of abstraction (such as coordinates or symbolic places) can be acquired from different sensors. This sort of semantics needs to be expressed and used in the process of activity recognition.

Researchers have applied classic machine learning techniques in the area of activity recognition. These techniques are sophisticated themselves, and are able to achieve good performance, while they exhibit a limited capability in using the domain knowledge and semantics underlying sensor data in their learning process. This drawback might make them consume a large amount of training data to build their structures or estimate parameters [3].

This paper will propose a new data structure, called a *context lattice*, to recognise human activities in a smart home environment. It is towards a formal model to represent and exploit semantics in pervasive computing. It allows expressing low-level semantics underlying environmental knowledge and sensor data and as well as deriving high-level semantics on situations; e.g., which situations a user cannot be involved at the same time. In this paper, situations are human everyday activities in a smart home environment, such as preparing a meal or watching TV. The low-level semantics will facilitate learning correlations between sensor data and situations, while the high-level semantics will help to guide situation-aware application design.

In earlier work we showed how context lattices could be used as a tool for understanding in-office hu-

*Corresponding author.

man activities, such as "in a meeting", "busy working at desk", or "coffee break" [4]. The context lattice was manually constructed by organising abstracted sensor data from positioning sensors, online calendars, and users' computer activity sensors, and labelling them with the pre-defined activities. The preliminary evaluation results were promising in expressing semantics and inferring activities. To ease the use of context lattices, we proposed a new approach that supports automatically generating context lattices and learning relationships between sensor data and activities [5]. This approach has performed well in a more complicated data set where a larger number of sensors are involved and relationships between sensor data and activities are less explicit. This paper extends previous work, tests the general applicability of context lattices in the area of activity recognition, and conducts comparative evaluations on two independent public real-world data sets.

This paper is organised as follows. Section 2 reviews the techniques that are widely used in the area of activity recognition and introduces the qualitative comparison between context lattices and the existing techniques. Section 3 describes the theoretical framework of context lattices. Section 4 introduces the data sets used for evaluation: the PlaceLab [6] and TK26M [7] data sets, and highlights their applicability for testing new approaches for human activity identification. Section 5 illustrates how a context lattice is constructed from scratch taking the PlaceLab data set as a main example. Section 6 introduces the activity recognition technique in a context lattice, and describes the experiments conducted on the context lattices built in Section 5, followed by a comparative analysis on the performances of context lattices. Section 7 discusses the knowledge extraction from a context lattice, including the human activity pattern, sensor performance, and relationships between situations. Based on our experience, we discuss the strength and limitation of context lattices in terms of theoretical contribution and practical usage in Section 8. Section 9 ends with a summary of our current work and suggests the future research direction.

2. Related Work

There are two types of approaches in the area of situation identification or activity recognition: rule-based [3,8,9,10,11], and learning-based approaches [4,6,12, 13,14,15,16,17].

2.1. Rule-based Approaches

A rule-based approach uses expert knowledge to define situations in logic statements. Earlier attempts focus on defining situations, including the work of Dey et al [18], Henricksen et al [10], and Wang et al [19]. They took the initiative in considering situations as the abstraction of sensor data that would be more influential on applications. They have worked on how to define situations in simple logical formulas using the pure expert knowledge. This work has been advanced by Loke [9], Costa et al [20], and Yao et al [11].

Loke [9] presents a declarative approach to representing and reasoning with situations at a high level of abstraction. A situation is characterised by imposing constraints on the output or readings returned by sensors. These constraints are represented as a logic program. This approach is based on the logical programming language LogicCAP that embeds situation programs in Prolog, which provides a high level of programming and reasoning with situations for the developers. The logical theory makes it amenable to formal analysis, and decouples the inference procedures of reasoning about context and situations from the acquisition procedure of sensor readings. This modularity and separation of concerns facilitate the development of context-aware systems.

Yau *et al* [11] and Costa et al [20] elaborate the natural characteristics of situations, including the relationship between context and situations, the composition of situations, and the temporal characteristics of situations. Yau *et al* [21] work on a composite definition of situations; that is, a definition of one situation can be composed of definitions of others. They also introduce time constraints in their definitions, including time stamp and interval, which are useful in studying the temporal sequence between situations. They have developed situation ontologies to facilitate the use of these formal representations.

Instead of dealing with individual situations, Thomson *et al* [3] provide a reusable library of situation definitions that helps to automatically determine situations. They express different levels of granularity of a situation through definition inheritance. Definitions for new situations are created as variations of existing ones so that the same situation can be interpreted at different levels of abstraction.

2.2. Learning-based Approaches

With learning-based approaches, developers apply off-the-shelf machine learning techniques, including

decision trees [14,6], Bayesian inference [22,23,4,24], Hidden Markov Models (HMMs) [25,26,27,16,17], and Conditional Random Fields (CRFs) [28,7]. All these works have presented good results on learning human activities from a limited number of advanced sensors (mostly on-body motion sensors). The following will briefly introduce the applications of these techniques in the area of activity recognition.

Bao *et al* [14,29] use decision trees to learn user body motions (such as bicycling, shaking hands, or typing) from the raw sensor data provided by accelerators on the human body. Van Kasteren *et al* [23] carry out activity recognition using a Bayesian framework. They use a static Bayesian model to learn the relationship between different sensor data and human activities, and also use a dynamic Bayesian network to model the temporal aspects of activities. Bui *et al* [22] use a multi-layer Bayesian dynamic structure, called an Abstract HMM, to track an object and predict its future trajectory in a wide-area environment. This structure is used to explicitly encode the complex and scalable spatial layout; i.e., the hierarchy of connected spatial locations. Trained with coordinate-based location data, it can predict the evolution of the object's trajectory at different levels of detail.

Minnen *et al* [27] and Wojek *et al* [16] used a layered HMM representation to infer office activities such as giving a presentation or making a phone call from the low-level sensor data including audio and video sensors and on-body motion sensors. Modayil *et al* [30] use interleaved HMMs to recognise multi-tasked activities where a person switches frequently between steps of different activities such as making a stir-fry, making a jello, and drinking a glass of water. The interleaved HMM records the last object observed from wrist-worn RFID sensors for each activity as a hidden state.

Liao *et al* [28] employ CRFs to construct models of high-level activities such as work, leisure, and visit. They use a person's GPS data to learn his activities over a few weeks, and then determine the relationship between the activities and places that are important to this person.

2.3. Summary

In summary, rule-based approaches are good at expressing knowledge, while their performance is undermined by the features of data in pervasive computing environment such as the complexity of knowledge and high uncertainty of sensor data. Since the knowledge can be about sensors, an environment (e.g., a space map), or users (e.g., a social network or tasks), developers must consider all of this and then be able to provide good definitions of situations. Also these approaches do not have a capability to resist noisy sensor data without the help of other techniques.

Learning-based approaches are good at automatically recognising activities, and resolving uncertainty of sensor data, while it is difficult to incorporate domain knowledge in their learning process and extract more advanced knowledge from them like high-level semantics on situations.

A context lattice takes the advantages of both rule- and learning-based approaches, and compensates for their deficiencies. It uses a semi-learning method in that it allows experts to express their knowledge about local domains as low-level semantics, such as the characteristic values of sensor readings and spatial relationships between locations. It enables to integrate pieces of the knowledge on sensor data, users, and environments, and learn the correlations between sensor data and activities; that is, what sensor data contributes to recognising an activity. The underlying assumption is that experts have more accurate knowledge on these local domains than their knowledge on correlating parts of local views. Therefore, a context lattice will result in less bias and subjectivity in utilising expert knowledge. Based on such low-level semantics, it can learn the correlations automatically through a training process and derive high-level semantics. The learned correlations enable it to resolve uncertainty to a certain degree.

3. Theoretical Work

Taking inspiration from lattice theory [31], we define a context lattice, to study the relationship between situations and context predicates that are characteristic functions on sensor data. A context lattice is made up of a set of nodes, each of which represents a logical expression and has a set of situations. A logical expression takes context predicates as input and applies logical operators on the predicates. A node is *activated* if its logical expression is satisfied by the current sensor data. The semantics of each node is that when a node is activated, any situation in its situation set can occur. Alternatively, any situation that is outside this situation set cannot possibly occur.

Figure 1 presents a simplified context lattice. There are two types of nodes in a context lattice: *prelimi-*

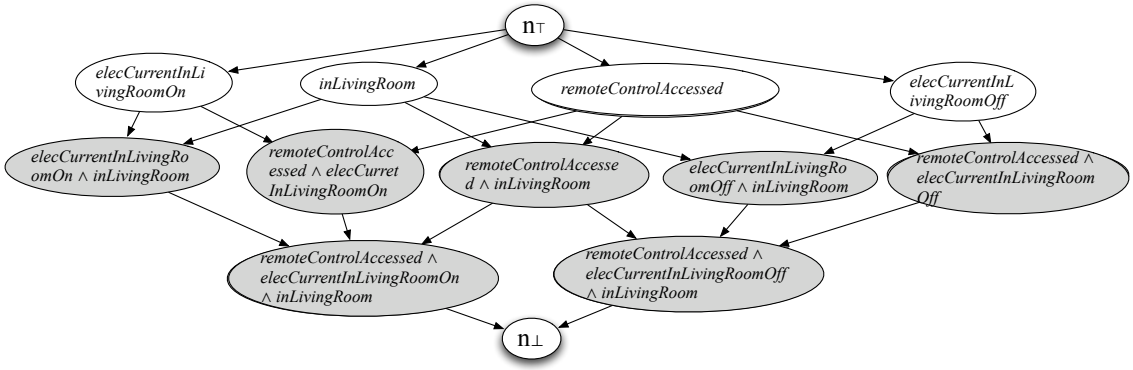

Fig. 1. A simplified context lattice. Except for the top and bottom nodes, other nodes are labelled with their logical expressions. Nodes in white are preliminary nodes while nodes in grey are compound nodes. No situations have been associated with the nodes in this context lattice yet.

nary nodes, whose logical expression is a single context predicate (e.g., `inLivingRoom`); and *compound nodes*, whose logical description is on a combination of context predicates (e.g., `remoteControlAccessed` $\wedge$ `inLivingRoom`). Nodes are organised with the *specialisation* relationship. A node is considered more *specific* ($\sqsubseteq$) than another node, if and only if the logical expression on the former node entails that on the latter node. The semantics of the specialisation relationship is that a node will be activated if and only if all its more general nodes are activated. The formal definition of a context lattice is given as follows.

Definition 1. *A context lattice is defined as a join semi-lattice $L = (N, \sqsubseteq)$, where N is a set of nodes and $\sqsubseteq$ is the partial order between the nodes.*

- *$\forall n \in N$, n has a logical expression denoted as $\mathbf{n}.\mathbf{l}$, and n corresponds to a set of situations denoted as $\mathbf{n}.\mathbf{S}$.*
- *$n_i \sqsubseteq n_j \in N$ iff $n_i.l \vdash n_j.l$, where $\vdash$ is the logical entailment relationship.*
- *$\forall n_i, n_j \in N$, there exists the unique node $n_k \in N$, $n_i \sqsubseteq n_k$ and $n_j \sqsubseteq n_k$ such that given any $n' \in N$, $n_i \sqsubseteq n'$ and $n_j \sqsubseteq n'$, $n_k \sqsubseteq n'$.*

The specialisation relationship between the nodes also uncovers a partial order on their situation sets. The activation of a node is the necessary condition that its more specific nodes will be activated. Thus it is impossible that situations that cannot occur on this node will occur on its more specific nodes. This result is formalised in Lemma 2.

Lemma 2. *If $n_i \sqsubseteq n_j \in N$, then $n_i.S \subseteq n_j.S$.*

We specify a few assumptions on these nodes, which are given as follows:

- *Assumption (A)*: there exists a unique top node $n_\top$ whose logical expression is a tautology `TRUE`. $n_\top$ corresponds to all situations. Its semantics is that if no sensor reading is available, then any situation is possible to occur;
- *Assumption (B)*: there exists a unique bottom node $n_\bot$ whose logical expression is a contradiction `FALSE`. $n_\bot$ corresponds to an empty set of situations. Its semantics is that if sensor readings are conflicting, then no situation can be correctly derived;
- *Assumption (C)*: no two nodes share the same logical expression.

A context lattice is a join semi-lattice. For any two nodes, there exists a join node that contains the greatest logical expression that is entailed by any logical expression on these nodes. For example, the node `inLivingRoom` is the join of the node `remoteControlAccessed` $\wedge$ `inLivingRoom` and the node `elecCurrentInLivingRoomOff` $\wedge$ `inLivingRoom`. The situation set on their join is the least superset of the union of their situation sets.

There does not necessarily exist a meet for any two non-conflicting nodes. If there exists a meet for a set of nodes, the logical expression on the meet contains the least logical expression that entails any logical expression on these nodes. For example, the node `remoteControlAccessed` $\wedge$ `inLivingRoom` is the meet of the node `remoteControlAccessed` and the node `inLivingRoom`. The situation set on their meet is the greatest subset of the intersection of their situation sets.

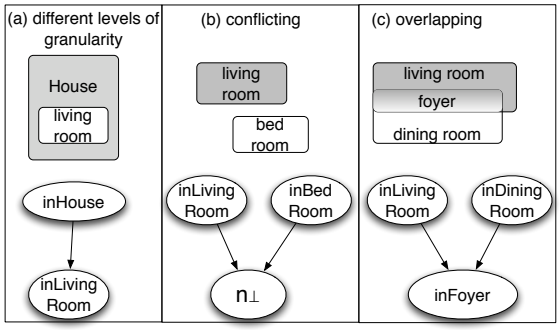

Fig. 2. Semantic relationships between context predicates [32]

Within a context lattice, we can express the low-level semantics through relationships between context predicates: various levels of granularity, conflicting, and overlapping (as seen in Figure 2). One context predicate is finer-grained than another if any sensor data that satisfies the former context predicate also satisfies the latter one. For example in Figure 2(a), the context predicate `inLivingRoom` is finer grained than the predicate `inHouse`, since the living room is contained in the house. According to Definition 1, various abstraction levels between two context predicates are represented as a specialisation relationship between two preliminary nodes that host these context predicates respectively.

Two context predicates are conflicting if any sensor data that satisfies one of them cannot satisfy the other. For example in Figure 2(b), the context predicate `inLivingRoom` is conflicting with the predicate `inBedRoom`, since they are spatially disjoint. The meet can be used to express a conflicting relationship. If two nodes host conflicting context predicates, then their conjunction is `FALSE`. According to *Assumptions (B)* and *(C)*, there exists a unique node $n_\perp$ whose logical predicate is `FALSE`, so the meet of the two nodes is $n_\perp$. This conflicting relationship is formalised in the following lemma.

Lemma 3. $n_i \sqcap n_j = n_\perp$, *iff* $n_i.l \wedge n_j.l = \mathit{FALSE}$.

Once the conflict relationship is defined between two preliminary nodes, it will be "inherited" by their more specific nodes. That is, if two nodes n_i and n_j are conflicting, then any of their more specific nodes n_i' and n_j' will conflict with each other (see Corollary 4). This will be used to avoid inconsistency (no compound nodes are created from conflicting nodes) during the process of automatically generating a context lattice.

Corollary 4. *If* $n_i \sqcap n_j = n_\perp$, *then* $\forall n_i' \sqsubseteq n_i$ *and* $\forall n_j' \sqsubseteq n_j$, $n_i' \sqcap n_j' = n_\perp$.

Two context predicates are overlapping if they can be satisfied at the same time by certain sensor data, but there exists sensor data that satisfies one of them but not the other. For example in Figure 2(c), the context predicate `inLivingRoom` overlaps with another predicate `inDiningRoom`, since they share a common location - the foyer. If two context predicates are overlapping, the more specific node under their corresponding nodes is the node with their overlapped logical expression; e.g., `inFoyer`.

This section shows that a context lattice is built on basic concepts of lattice theory. With its structural features, a context lattice can express these typical semantic relationships of context predicates. More details on the theorectical work of the context lattice can be found in [4,5].

In the literature of applied Lattice theory, it has been used in the formal data analysis, knowledge representation, and information management [33,34,35]. A *concept lattice* is designed to organise formal concepts, each of which is constitued by its extension (i.e., a collection of objects belonging to a concept) and intension (i.e., a collection of attributes common to all those objects). The concept order is based on a coupled extensional and intensional relation. In a context lattice, a node can be considered as a concept, where its logical expression can be regarded as an intension of the concept and its situation set as an extension of the concept. The difference is that the context lattice uses a lighter theoretical foundation of Lattice theory, and puts more emphasis on learning and reasoning on the association between the logical expressions and situations.

4. Data sets in Smart Home Environments

To test the general applicability of the context lattice, we demonstrate its feasibility and evaluate its performance in activity recognition on two independent real-world data sets – the PlaceLab [6] and the TK26M data set [7].

4.1. PlaceLab Data set

The PlaceLab is an instrumented home that contains over nine hundred sensors. The PlaceLab data set [1]

[1]The data set can be downloaded at: `http://web.media.mit.edu/~intille/data/PLCouple1/`.

(also known as PLCouple1) was gathered over a period of 15 days that a married couple (who were unaffiliated with the PlaceLab research) lived in the PlaceLab. During this period, they were encouraged to maintain their life routine as normally as possible. The PlaceLab was instrumented with the audio-visual recording infrastructure that was used to record activities of the subjects except for private activities (such as bathing). The video was annotated by a third party, which provided a ground truth representing the activities that were taking place over the period of study. So far, only the activities of the male subject have been annotated.

The diary shows that most activities annotated were occurring between the hours of 17 and 24. The human activities include "using phone", "using a computer", "reading", "eating", "meal preparation", "watching TV", "dishwashing", "hygiene", and "grooming". The occurrences of these activities vary greatly; for example, "using computer" occurs for 20.14 hours, which covers 42.9% of the data set, while "dishwashing" occurs only for 9.83 minutes, covering 0.35%.

Besides the sensor data and ground truth, the PlaceLab data set contains a location map of the home and a sensor metadata file [2], which records the meta information about each sensor input, including its type, its identity (ID), where it is installed, and which object it is attached to; e.g., the couch in the living room. The technical specification of the sensors was published making it possible to interpret the raw sensor data. A data set visualiser was also made available, called HandLense, which we used to observe the behaviour of sensors in the environment and define critical output values for sensors by examining the sensor readings for each sensor ID in the sensor output files.

Our experiment covers most sensors in the PlaceLab data set, which includes wireless infra-red motion sensors, "stick-on" object motion sensors, switch sensors, RFID, and electrical current, water, and gas flow sensors. We do not use environmental sensors (like the sensors to detect temperature, humidity, and barometric pressure) since they do not have a direct effect on human behaviours that we intend to detect in this paper. This data set includes the on-body 3-axis motion sensors that provide the acceleration data of the subject's movement. If there existed a diary about fine-grained activities that recorded the specific movement of the subjects, such as walking, sitting, moving the arm, or lifting an object, it would be possible for us to train the sensor data to identify these fine-grained behaviours using supervised machine learning techniques [29]. However, since the PlaceLab data set only provides a diary that records higher level activities, such as "grooming" or "watching TV", we cannot define movement predicates from these accelerometers (and also training the body motion sensors is not in the scope of our work). Thus, we do not use these sensors in our experiments.

[2]The sensor metadata file can be downloaded at: `http://web.media.mit.edu/~intille/data/PLCouple1/PLObjects_oct23.xml`

4.2. TK26M Data set

The TK26M data set was gathered by researchers in the intelligent autonomous systems at the University of Amsterdam [7]. It was built on ordinary activities over 28 days in a house where a 26-year-old man lived. The house consists of five rooms, in which are installed 14 state-change sensors on the doors, microwave, fridge, freezer, washing machine, or cupboards. When a sensor was fired, it was recorded in the data set. The activities were annotated by the participant through a Bluetooth headset in real-time. They include "leave house", "use toilet", "take shower", "go to bed", "prepare breakfast", "prepare dinner", and "get drink".

5. Construction of a Context Lattice

A context lattice is constructed through the following processes: (1) defining primitive nodes from context predicates that are characteristic functions on raw sensor data; (2) creating compound nodes based on low-level semantics between context predicates; (3) associating situations with each node in the context lattice through a training process. The Java API has been implemented to allow developers to construct a context lattice easily. The following section demonstrates the construction process from scratch with the use of the Placelab data set.

5.1. Defining Context Predicates

To create a context lattice, context predicates should be abstracted from sensor data and then preliminary nodes will be generated from each single context predicate. A context predicate is defined with a set of sensor IDs and a constraint on readings of these sensors. A context predicate holds if its constraint is satisfied by a reading from any of these sensors.

From the PlaceLab data set we classify the sensor IDs according to the types of sensors, and define context predicates for each sensor ID. Mappings from raw sensor output to context predicates are defined based on the sensor's technical specification and observation of sensor behaviour. For example, a context predicate `doorInBedroomOpen` is defined as true when the switch sensor on the bedroom door produces readings 0 or 200 as per the technical specification. The sample code is shown as follows. The lines 1-3 create an atom formula that consists of a sensor type "SWITCH", a sensor ID "2F00000022B46D12Y", and a characteristic function on its reading "%200 == 0". The lines 4-6 define a context predicate with a descriptive name and the atom formula.

```
1. Formula af = new AtomFormula
2.      ("SWITCH", "2F00000022B46D12Y",
3.      "%200 == 0"));
4. ContextPredicate cp = new
5.   ContextPredicateImp("doorInBedroomOpen",
6.                       af);
```

Another predicate `inLivingRoom` is defined as being true when the sensor reading for the infra-red motion sensor in that room is no less than 5, based on observations using the PlaceLab visualiser – HandLense. Using the same approach, context predicates are defined for the electrical current, water, and gas flow sensors. Since sensors in each type share the same characteristic function (e.g., the above formula on a switch sensor), developers only need to define a characteristic function for each type of sensors, and write a script to parse the PlaceLab sensor metadata file so as to automatically generate context predicates for each sensor.

The time context predicates are also defined since the activities are potentially time-related; e.g., the subject usually gets dressed or undressed (both of which are defined as "grooming") when he arrives home and before he goes to bed. The time predicates are defined as hourly segments between the hours of 17:00 and 24:00, e.g., `17-18` or `23-24`. `TimeFormula` is used to create and evaluate temporal related formulas. In the following example, a context predicate `17-18` is defined on a time formula that evaluates whether the input time's hour is 17.

```
1. Formula tf = new TimeFormula
2.     ("hour", "== 17"));
3. ContextPredicate cp = new
4.     ContextPredicateImp("17-18", af);
```

Certain objects in the PlaceLab have multiple sensors relating to their use; e.g., a mobile object might have both RFID and object motion sensors associated with it. When this occurs, we combine their outputs directly by linking all sensor readings from a single object to a single predicate. For example, an object access predicate `laptopAccessed` is defined on the laptop that has two sensors associated to it: its RFID sensor "E00700001E226FEE" and its motion sensor "956". The predicate holds if either sensor gives an active reading. The following sample code shows that a context predicate can be defined on a disjunction of atom formulas, which implies that if either the RFID tag is read or the object motion sensor is fired, then *laptopAccessed* holds.

```
1. LogicConnection<Formula> afs
2.     = new Disjunction<AtomFormula>();
3. afs.add(new AtomFormula("RFID",
4.      "E00700001E226FEE", ""));
5. afs.add(new AtomFormula("OM",
6.      "956", ">= 0"));
7. ContextPredicate cp = new
8.   ContextPredicateImp("laptopAccessed",
9.                       afs);
```

When it comes to more complex sensors, machine learning techniques can be used to define context predicates in that a context lattice does not support learning characteristic values on numeric sensor data. For example, if a machine learning technique extracts a characteristic function on sensor data from wearable accelerators, then body motion predicates like `stand` or `walk` can be defined.

As mentioned in Section 3, developers can apply domain knowledge to define the low-level semantics. The domain knowledge can be about an environment such as a space map as shown in Figure 2. In the PlaceLab environment, the location predicates conflict with each other, since each of them represents an individual room and the subject cannot be in two different rooms at the same time. Relationships between predicates can also be evaluated from their characteristic functions; e.g., the predicates `lightInLivingRoomOn` and `lightInLivingRoomOff` cannot hold at the same time since they have exclusive range of active readings. The relationships between the context predicates can be specified by the developers (e.g., the location predicates) or evaluated quantifiably (e.g., the light predicates). More details have been discussed in [36].

So far the preliminary nodes have been defined on different types of sensor data. To complete a context lattice, these preliminary nodes are needed to be com-

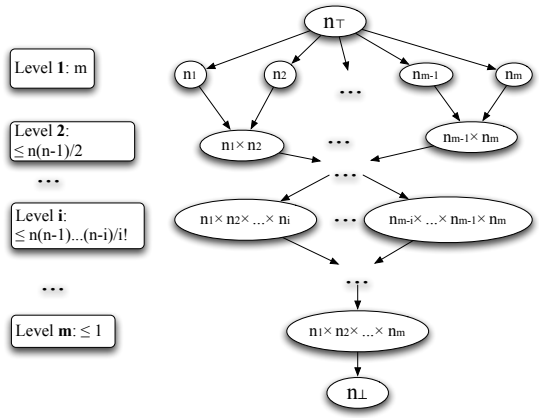

Fig. 3. The automation process to generate compound nodes.

bined iteratively, which will be described in the following section.

5.2. Combining Context Predicates

A context lattice will be automatically completed by generating compound nodes from preliminary nodes, level by level, until all the non-conflicting nodes are combined. At the current stage, a logical expression on a compound node is implemented as a conjunction of context predicates. Its creation process is described as follows. Given two nodes n_i and n_j, their compound node (labelled as $n_i \otimes n_j$) can be

1. the bottom node, if they conflict, which is evaluated by using Lemma 3;
2. or a new node $n_{ij} = n_i \otimes n_j$ with
   - $n_{ij}.l = n_i.l \wedge n_j.l$;
   - $n_{ij} \sqsubseteq n_i$ and $n_{ij} \sqsubseteq n_j$.

As shown in Figure 3, if all non-conflicting nodes were combined, the complexity of generating a context lattice would be $O(2^N)$, where N is the number of preliminary nodes. This complexity would give rise to a massive scalability problem, which keeps a context lattice from being meet complete. A practical solution is to combine nodes if their predicates are not conflicting and closely related to each other. The underlying assumption is that the context (or the sensor data) closely related to the surrounding of the subject will have a major effect on inferring his current situations. This context lattice has been configured to combine context predicates if the sensors they refer to are static and physically located in the same location. For example, the node `elecCurrentInLivingRoomOn` is combined with the node `lightInLivingRoomOn`, while it will not be combined with the node `elecCurrentInOfficeOn`. This solution is reasonable if activities do not transcend the boundaries of a single location; e.g., the subject cannot be cooking in both the kitchen and bedroom.

Seven location predicates have been defined for each individual room, one for each room in the PlaceLab, and seven time predicates, one for each hour in the data set. All the sensors are divided into two groups: the sensors that are statically located in one of the seven locations; and the sensors attached to mobile objects that are independent of location. Static sensor predicates cannot be combined with static sensor predicates from other locations but mobile object access predicates (e.g., `laptopAccessed`) can be combined with any other sensor predicates. The classification is done with the use of the sensor metadata file; that is, each context predicate is grouped according to its corresponding sensor's location recorded in the sensor metadata file.

For each location, the generation algorithm automatically creates compound nodes by combining the non-conflicting time, location, and object access predicates. For example in the living room, there are four current predicates with two pairs of conflicting predicates. With these current predicates, there will be 8 compound nodes created. In the living room, 624 compound nodes are generated from the combination of these 8 compound current nodes, 7 time predicates, and 8 object access predicates ($8*7+7*8+8*8+8*7*8$). In the end, the completed context lattice has 27649 nodes in total: 1 top node, 1 bottom node, 113 preliminary nodes and 27534 compound nodes. This context lattice is easier for a system to manage, compared to a fully generated lattice with about 2^{113} nodes.

5.3. Labelling Nodes with Situations

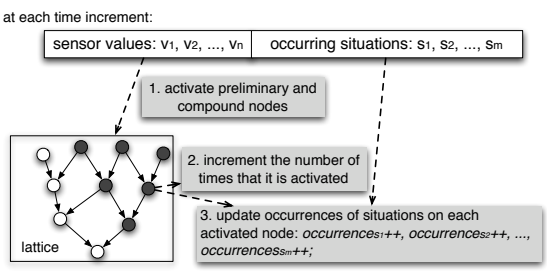

Fig. 4. A process of training a context lattice [5].

In order to link activities to context predicates in a context lattice we execute a training algorithm that consists of three steps, which are illustrated in Figure 4. Training data are passed into the raw context lattice, one time increment at a time, whose frequency depends on the data set; e.g., 10 seconds for the PlaceLab data set. At each time increment, the context lattice is fed with a set of sensor values v_1, v_2, ..., v_n and a set of activities s_1, s_2, ..., s_m, which are occurring at that time. The training process activates the preliminary nodes by evaluating their context predicates against the sensor values. The context lattice applies the meet operator on the activated preliminary nodes to activate their compound nodes. For each activated node, a variable is updated, which indicates how many times its logical expression has been satisfied. The number of times that each of the occurring activities occurs on the activated node will be incremented.

At the end of the training process, each node will record (1) the number of times that it has been activated; and (2) a situation occurrence array, containing the number of times each activity occurred on this node when it has been activated. Each entry in the situation occurrence array is normalised by being divided with the total number of times that activity occurs in the training data. The normalised value implies the probability that an activity is occurring when this node is active; if it is zero this activity is impossible to occur.

If a node's occurrence is zero or its situation set is empty, then this node is considered *ineffective* and will be removed from the context lattice. We synchronise all the annotated sensor data and diary data into 10-second gaps, which covers all the annotated 40 hours out of the total 104 hours' data in the PlaceLab data set. Two thirds of these synchronised instances are used to train the context lattice built in the above section. After training and pruning off ineffective nodes, the size of the context lattice decreases to 13035 nodes, which is close to half of the initial size. This means the training process can filter quite a few meaningless combinations of predicates.

6. Activity Recognition

When a context lattice is built and completed, it is ready to be used to recognise human activities. The structure of a context lattice makes forward chaining algorithms more efficient by evaluating all the predicates just once. When fed with a set of sensor readings, the system starts by evaluating the predicates on all the preliminary nodes from the top. Activity recognition is a procedure of applying the meet operator on all the activated preliminary nodes. Since the context lattice is not meet-complete as stated in Section 5, the procedure may end up with a set of nodes (called *resultant nodes*) which are most specific among all the activated nodes, rather than a unique node.

Instead of inferring a single activity, a context lattice will return the following two sets of situations where each situation is provided with its occurrence ratio: (1) a set of *possible-to-occur* situations that is the union of the situations in these resultant nodes, which indicates that the situations outside this set are impossible to occur; and (2) a set of *most-likely-to-occur* situations that is the intersection of the situations in these resultant nodes, which indicates that these situations are agreed by all the sensor data.

For example, if a context lattice ends up with two resultant nodes whose situation sets are {"using computer", "reading"} and {"watching TV", "reading"} respectively, then the activities possible to occur are in {"watching TV", "using computer", "reading"}, and the activities most likely to occur are in {"reading"}.

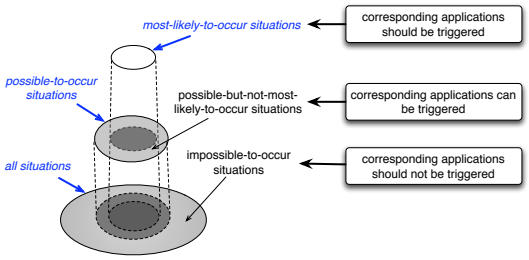

Fig. 5. Relationships between applications and inferred situations.

The inference result of a context lattice provides the accurate and detailed information about what is occurring in the real world. This will help application developers to design a more robust and customised system compared to the techniques mentioned in Section 2. Figure 5 presents the relationship between recognised activities and confidence of carrying out corresponding applications. Applications (or services) related to the most-likely-to-occur situations will be provided with a high confidence, while applications whose corresponding situations are not contained in the possible-to-occur situations should never be triggered. Applications related to the situations that are possible but not most likely to occur are not suggested to be triggered, but they can be triggered with the consideration of other design requirements.

6.1. Evaluation Methodology

The activity recognition will be evaluated on both the PlaceLab and TK26M data sets. The evaluation parameters are *precision* and *recall*. Precision is the ratio of the times that a situation is correctly inferred to the times that it is inferred in most-likely-to-occur situation sets. Recall is the ratio of the times that a situation is correctly inferred in possible-to-occur sets to the times that it occurs.

When measuring recall, it is possible to achieve 100% recall under certain circumstances. If there is significant conflict between sensor data, the most-likely-to-occur situation set could be empty, and the possible-to-occur situation set could contain every situation. This leads to 0% precision and 100% recall, which suggests that all the situations are equally possible to occur, but none of them is most likely to occur. This makes both the possible-to-occur and the most-likely-to-occur situation sets insensitive to sensor data.

To avoid this insensitivity, thresholds are set on occurrence ratio of each situation to refine situations in the most-likely-to-occur and the possible-to-occur sets. The thresholds are determined with the standard *maximum likelihood estimation*; i.e., a threshold is chosen if it produces the highest *F-measurement* that is a standard measurement weighting precision and recall equally: $F = \frac{2 \times precision \times recall}{precision + recall}$. It will be used to measure the overall accuracy in the following section. When the conflict is detected in the activated nodes, the most-likely-to-occur situation set will contain the situations with the highest probability, which excludes the possibility of 100% precision. The possible-to-occur situation set will filter out the situations whose occurrence ratio is below its corresponding threshold, which decreases the possibility of 100% recall but does not exclude it. Therefore, to test the existence of insensitivity the chance that a possible-to-occur situation set covers all the situations will be calculated.

We choose thresholds by using the *maximum likelihood estimation*, where the threshold is tuned to provide the best inference result. For each situation, we set in initial threshold and a scale to reduce the threshold to 0. For each situation, we calculate the precision and recall for each threshold. In the edn, for each situation, we will choose the inference result that achieves its highest precision and recall.

The performance of the context lattice is evaluated using stratified 10-fold cross validation, which is considered standard in the machine learning community [37]. Three types of experiments will be conducted on the context lattices. First of all, we will test the existence of insensitivity of context lattices, and measure their precision and recall. The precision and recall will be compared to the results obtained from default configured J48 decision tree and Naive Bayes using the Weka software package [37]. The reasons that we choose them are that (1) they are off-the-shelf machine learning techniques, and that (2) they achieve better classification results on the PlaceLab data set than the other techniques [6]. These experiments will show that the accuracy of a context lattice in activity recognition is comparable with traditional machine learning techniques. We will conduct a second experiment to demonstrate that domain knowledge can be easily utilised in a context lattice and can be used to improve the learning process. The recognition accuracy will be evidently increased compared to the result in the first experiment. Our third experiment will evaluate the effect of the amount of training data on the inference result.

6.2. Experiments on the PlaceLab and TK26M Data Sets

The context lattice built on the PlaceLab data set is evaluated using the 13870 synchronised sensor and diary instances. There are 250 (1.8%) times that the possible-to-occur situation sets cover all the activities. It is shown that there only exists a very small chance that the possible-to-occur situation sets are insensitive.

Figure 6 and Figure 7 present the comparison of precision and recall of activity recognition between J48 decision tree, Naive Bayes, and the context lattice. From Figure 7, the context lattice produces the highest recall on most of the activities (except for "hygiene"), since the possible-to-occur situations contain all the possible activities that are above the threshold. It shows that the possible-to-occur situation set is good at capturing the activities that are possible to occur. This set will make applications more robust. In terms of the precision, it has a relatively lower performance of the precision of recognising activities on the PlaceLab data set.

The context lattice built from the TK26M data set is evaluated using the 796 synchronised sensor and diary instances, which are obtained in the same way as we did on the PlaceLab data set. Since no activities co-occur in this data set, the most-likely-to-occur situation set only infers one activity every time. Dur-

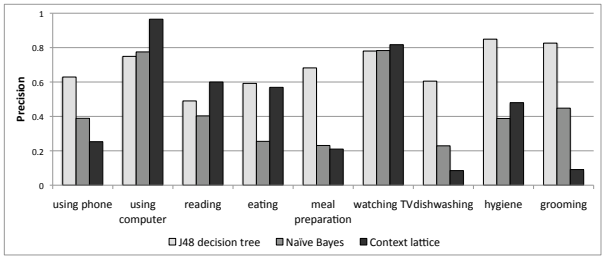

Fig. 6. Comparison of precision of inference using the PlaceLab data set between J48 decision tree, Naive Bayes, and context lattice.

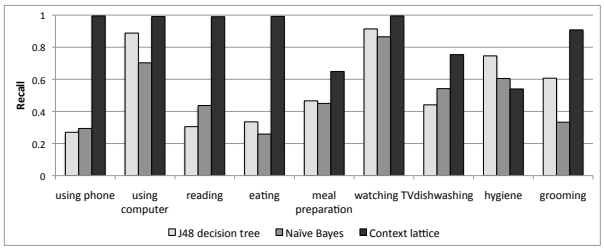

Fig. 7. Comparison of recall of inference using the PlaceLab data set between J48 decision tree, Naive Bayes, and context lattice.

ing this evaluation, the possible-to-occur situation set never covers all the activities; that is, the insensitivity never occurs.

Figure 8 and Figure 9 show the comparisons of precision and recall between J48 decision tree, Naive Bayes, and the context lattice on the TK26M data set. It is shown that the context lattice has better performance than the other two techniques.

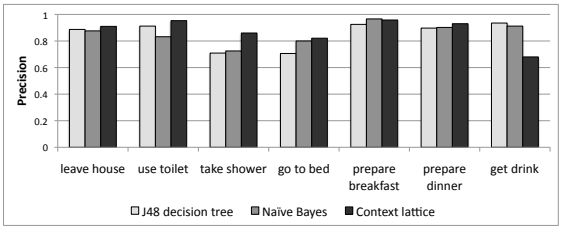

Fig. 8. Comparisons of precision using the TK26M data set between J48 decision tree, Naive Bayes, and context lattice.

6.3. Domain Knowledge

Figure 10 presents an overall accuracy on the PlaceLab and TK26M data sets between J48 decision tree, Naive Bayes, and the context lattices, which is measured in F-measurement. It shows that context lattices work less effectively on the PlaceLab data set than on the TK26M data set. To better understand the performance of context lattices, we analyse the characteristics of activities that are inferred in low precision on the PlaceLab data set.

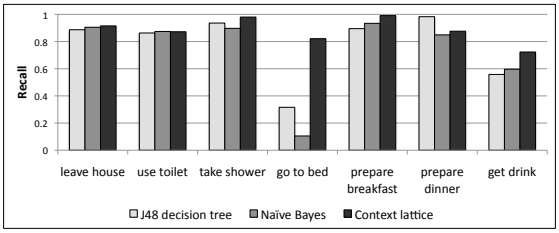

Fig. 9. Comparisons of recall using the TK26M data set between J48 decision tree, Naive Bayes, and context lattice.

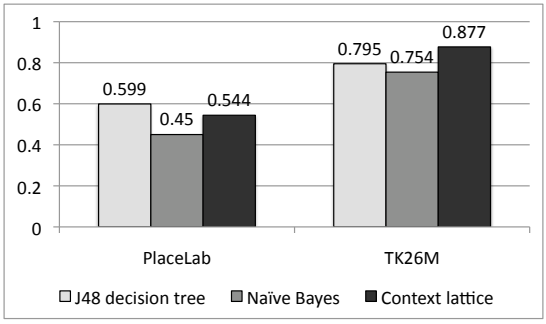

Fig. 10. Comparison of the averaged F-measurements between context lattices, J48 decision tree, and Naive Bayes on the PlaceLab and TK26M data sets

The precision of inferring an activity will be low if this activity can co-occur with any other activity or it does not have an explicit pattern detectable from existing sensors. For example, “using phone” can occur with any other activity at any time in any room as shown in Figure 6; and “get drink” frequently co-occurs with “prepare breakfast” and “prepare dinner” as shown in Figure 8. Since this activity cannot be identified or distinguished from other activity without the help of other characterised sensor data, it is co-inferred frequently with others, which results in low precision.

This case cannot be dealt within a context lattice, unless a new sensor effective in identifying the activities is introduced. The PlaceLab data set involves the on-body motion sensors to sense the acceleration of the movement of the male subject’s thigh and wrist. These sensors might be useful in distinguishing some activities such as “eating” and “reading” but would require separate training to interpret the meaning of the raw sensor data, as mentioned in Section 4.

The precision will be low if the sensors that are supposed to identify this activity produce inaccurate

data or are accidently or incorrectly fired by other subjects. For example, the activity "dishwashing" should be identifiable if the positioning sensor can accurately locate the male subject's location (i.e., in the kitchen), and the faucet in the kitchen can recognise that the male subject is using it. As mentioned in [38,6], there are two participants involved in the PlaceLab data collection and most sensors except RFID are not user-specific. Thus one subject could activate a sensor that should not be fired when the other subject is participating an annotated activity.

This case can be resolved to a certain degree by introducing further domain knowledge into the context lattice. That is, we will enforce the relationship between sensor data and activities, based on human common-sense knowledge and understanding of sensors. For example in an activity "dishwashing", we propose two assumptions:

- *Assumption 1*: it cannot occur in other locations except the kitchen. Since "dishwashing" is an activity that involves observable motions, if the infra-red sensor senses a motion in another location instead of the kitchen, then it is impossible that the male subject is washing dishes. This assumption would be more accurate if there is only one participant living in the house. In the Placelab data set there exists another participant, however, these two participants often did things together in the same location [6]. Thus, we consider this assumption still reasonable.
- *Assumption 2*: if the water flow sensor does not sense flowing water in the faucet in the kitchen, then "dishwashing" cannot occur.

These two assumptions aim to improve precision by filtering the sensor data to remove the cases where "dishwashing" was incorrectly trained. The assumptions can be manually specified by developers in the construction API, which restrict the possible situations on the corresponding preliminary nodes. That is, developers simply add one statement to certain preliminary nodes to enforce the occurrence of "dishwashing" to be always zero in their situation arrays. Thus, all these preliminary nodes are exclusive from "dishwashing" in the training process. For *Assumption 1*, all the preliminary nodes that associate with location predicates other than `inKitchen` will be enforced exclusive from "dishwashing". For *Assumption 2*, the preliminary node whose context predicate is `faucetInKitchenOff` is enforced exclusive from "dishwashing". These restriction statements on the preliminary nodes will be automatically "inherited" by their more specific nodes with respect to Lemma 2. A sample code is listed as follows. The preliminary node handler adds an exclusive situation whose name is "dishwashing" to a location predicate whose name is `inLivingRoom` (lines 1-2) and to a water flow predicate whose name is `faucetInKitchenOff` (lines 3-4).

```
1. PreNodeHandler.addExSitu(
2.       "inLivingRoom","dishwashing");
3. preNodeHandler.addExSitu(
4.      "faucetInKitchenOff","dishwashing");
```

Besides the situation "dishwashing", we will also deal with the situation "grooming". This situation could involve some miscellaneous like "brushing or styling hair", so we will only apply a simple assumption:

- *Assumption 3*: it cannot occur in the kitchen, or the office.

After adding these relationships to the original context lattice, we re-evaluate it in the 10-fold cross validation using the same data set. Figure 11 presents the comparison of the F-measurements between the original and improved context lattices. The F-measurement of "dishwashing" has been greatly increased, which is even better than J48 decision tree and Naive Bayes. The F-measurement of "grooming" has been increased, but not as much as that of "dishwashing", since "grooming" is less specific. For example, a "grooming" activity "brushing or styling hair" can occur anywhere at any time, which breaks *Assumption 3*.

The introduced assumption does not cause an apparent change in the inference result on the other activities. The overall precision has only been increased slightly, mostly because these two activities occur much less frequently compared to all the other activities.

The above example has demonstrated that applying a small amount of domain knowledge in a context lattice can greatly improve the accuracy of inference such that the uncertainty of sensor data has been resolved to a certain degree. In addition, the process of introducing domain knowledge is simple, which adds statements on the preliminary nodes for each assumption. A context lattice is configured to facilitate both the expression and propagation of the knowledge as described in Lemma 2 and Corollary 4 in Section 3.

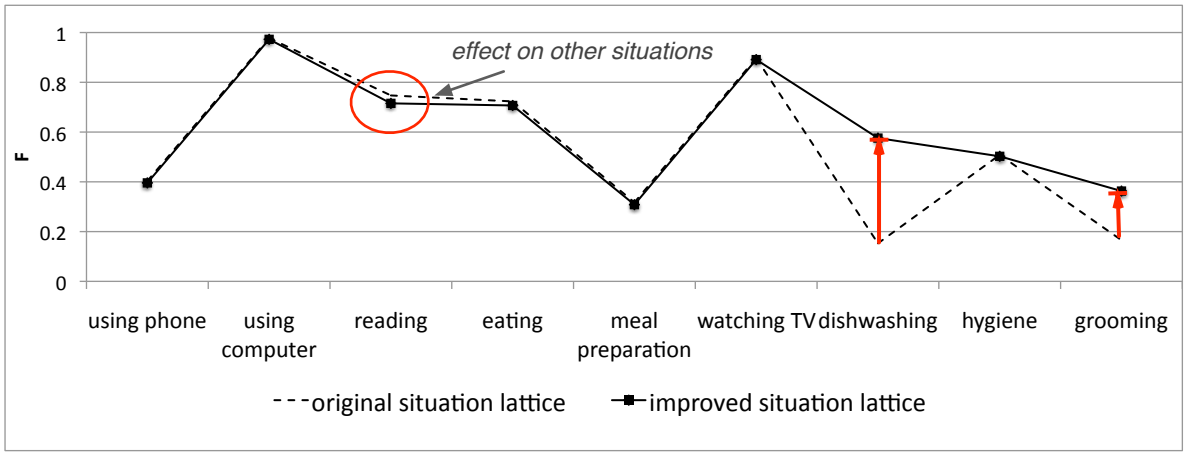

Fig. 11. Comparison of F-measurement between the original context lattice and the improved context lattice with domain knowledge.

6.4. Training Data

A context lattice needs to be trained with sensor and diary data to learn the semantic relationships between context predicates and situations. The following experiments are conducted to evaluate how the amount of training data will affect the precision and recall of recognising activities. Given n days' data, the experiments will evaluate the precision and recall by using the following amount of training data: 1 day, 2 days, and until $n-1$ days. For each number of days $m-1$, we choose each day's data for testing and choose the previous m days data for training. If the previous m days are unavailable, then the backward days' data will be chosen [7]. For example in the PlaceLab data set, for using one day's data for training, if the data on 2006-08-23 (the first date) are used for testing, the data on 2006-09-18 (the last date) will be chosen for training.

Figure 12 and Figure 13 present the relationship between the averaged F-measurements on all the activities and the amount of training data on the PlaceLab and TK26M data set.

The curves in both figures rise asymptotically, which means that the context lattices are insensitive to the amount of training data. Both figures show that using one day's data for training does not make much difference from using more days' data for training. Beyond a certain quantity the further addition of training data does not have a significant effect on the accuracy of inference. Figure 12 shows that using only half day's data for training and another half day's data for testing will reduce the performance, since the first half day's data might lack enough training data to cover the activities that occur in the next half day.

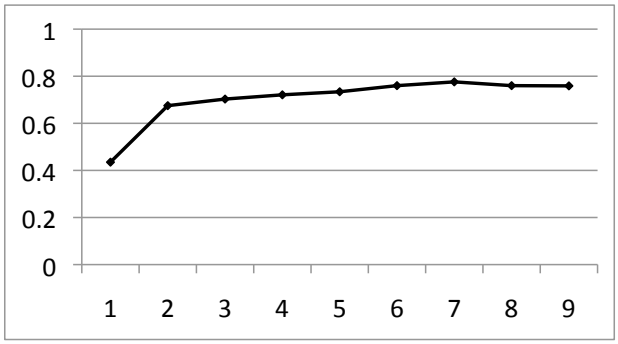

Fig. 12. Relationship of F-measurements and the amount of training data using the PlaceLab data set.

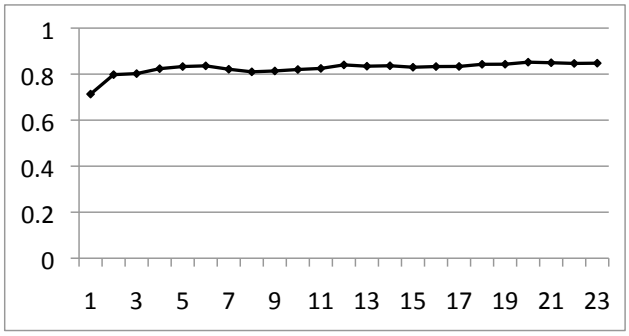

Fig. 13. Relationship of F-measurements and the amount of training data using the TK26M data set.

The reason that a context lattice consumes a small amount of training data is that it has provided a structure to organise context predicates, and the training

process only needs to label activities on different combinations of predicates. As long as the training data contain the correct mapping between activities and sensor data, then a context lattice will be able to finish the learning process. Training does not include the process of building a new hierarchy, which is the reason that makes most machine learning techniques (such as decision tree and Naive Bayes) consume much training data.

7. Knowledge Extraction

This section describes how to extract high-level semantics by observing characterised sensor data on situations; that is, how sensors behave when a user is performing a certain activity. This type of knowledge will be helpful in accumulating experiences on choosing sensors to build a future smart space, and providing feedback on designing situation-aware applications.

7.1. Situation Specifications

A context lattice supports deriving a specification for a situation. A situation's specification is a logical expression that takes context predicates as input and applies logical operators (disjunction and conjunction) on them. A situation is considered being conducted, if its specification is satisfied by current sensor data.

Given a built context lattice with N nodes, there exists a set of the most specific nodes (except the bottom node) $N_s \subseteq N$ whose situation set contains a certain situation s. The specification of this situation s is generated by applying the logical operator `OR` on the logical description of these nodes N_s. That is, a specification of s is $n_1.l \vee n_2.l \vee \ldots \vee n_m.l$, where $\{n_1, n_2, \ldots, n_m\} = N_s$.

We derive specifications of situations from the context lattice of the PlaceLab data set. Some of the situations have a relatively small number of nodes in that they have a tractable activity pattern. For example of the activity "hygiene", its specification[3] is listed as follows:

hygiene =(inPowderRoom ∧ 18-19
∧waterInPowderRoomOff
∧lightInPowderRoomOff
∧ nothingInPowderRoomAccessed)
∨(inBathroom ∧ 23-24
waterInBathroomOff ∧
lightInBathroomOn))

Some of the frequently-occurring activities are associated with a large number of the most specific nodes. For example, "using phone" has 245 most specific nodes, since the subject could use phone anywhere at any time. Part of its specification is listed as follows:

using phone =inLivingRoom ∧
((elecCurrentInLivingRoomOff∧
nothingInLivingRoomAccessed ∧
16-18) ∨ (lightInLivingRoomOn
∧ (((18-19 ∨ 20-21)
∧ remoteControlAccessed)
∨ (nothingInLivingRoomAccessed
∧ 18-20))))
...

By examining the specifications of the activities, developers can have an intuition on what sensors are useful in recognising an activity. For the activity "hygiene", the most effective sensor is a positioning sensor that could detect the subject in the powder room or the bathroom. Also the number of most specific nodes on each situation can reflect the accuracy of its descriptive definition; that is, a large number of nodes suggest that either this situation occurs frequently (e.g. the situation "using a computer" has 395 most specific nodes, and it occurs 42.9% of the time through the data set) or this situation is less tractable. Under the latter circumstance, developers might need to refine its descriptive definitions by removing trivial activities (like styling hair) or split it into sub activities.

The specifications of the activities cannot only be represented in the above logical rules, but also can be visualised in a partial lattice that only includes all the nodes associated with a certain situation. Since the nodes have the *specialisation* relationship in the context lattice, it might be easier for understanding and diagnosing [39].

7.2. Semantics between Situations

Based on specifications of situations, we can explore their semantic relationships: *type-of* and *conflicting*. These two relationships can be inferred from the specifications of situations based on their correspondingly most specific node sets. One situation s_i is a type of another situation s_j, if when a subject is conducting s_i he is also considered conducting s_j. It requires that for any of the most specific nodes on s_i, there exists at lease one most specific nodes on s_j such that the

[3]The specifications presented in this section are re-organised by authors for the sake of readability and space.

node on s_i is more specific than the node on s_j (as presented in Definition 5). Thus if a specification of s_i is satisfied, then the specification of s_j will be satisfied as well.

Definition 5. *s_i is a type of s_j, iff* $\forall n_k \in N_{s_i}, \exists n'_k \in N_{s_j}, n_k \sqsubseteq n'_k$.

One situation s_i conflicts with another situation s_j, if it is impossible for a subject to be in both situations at the same time. It requires that any of the most specific nodes associated with s_i conflicts with any of the most specific nodes associated with s_j (as presented in Definition 6).

Definition 6. *s_i conflicts with s_j, if* $\forall n_k \in N_{s_i}, n_l \in N_{s_j}, n_k \sqcap n_l = n_\perp$.

If two situations share some of the same most specific nodes, then they are likely to occur at the same time when any of these nodes are activated. However, this leads to another question: if one of these nodes is activated, it is possible that the user is in both situations occur, or the user is one of them not the other. For example in the PlaceLab data set, "watching TV" and "eating" can share the same most specific node, whose context predicates are `19-20` $\wedge$ `inLivingRoom` $\wedge$ `elecCurrentInLivingRoomOn` $\wedge$ `nothingInLivingRoomAccessed`. When this node is activated, the subject can be "watching TV" and "eating" simultaneously, or he can be either "watching TV" or "eating". This issue is related to the discernability of sensors in precisely identifying situations, which has been covered in more details in [5].

We apply the above two definitions on the context lattice of the PlaceLab data set. There are no specialisation relationships between the activities. Since the location predicates on each individual rooms conflict with each other, the activities that occur in different rooms should conflict, such as "dishwashing" and "hygiene". Due to the imprecision of the positioning sensors, the activities that should be conflicting are not inferred as conflicting; for example, "watching TV" should conflict with "meal preparation".

8. Discussion

This section discusses the strength and limitation of context lattices and propose potential improvements on them from the perspectives of theoretical work and practical usage.

8.1. Capability in Expressing Logic

Theoretically, a context lattice should be able to support any logical formula as long as the specialisation relationships exist between the nodes. Due to the scalability issue of the lattice and the complexity of implementation, we only support conjunction and disjunction in the logical expressions at the current stage. However, each context predicate can implicitly involve universal or existential quantification and implication; for example, a location predicate `inLivingRoom` would be, if translated in logic, $\exists$ `sensor`

```
(sensor.id = "2088" AND sensor.reading > 0
  -> inLivingRoom = TRUE) .
```

The API is provided to define simple predicates, to generate a conjunction of these predicates on compound nodes, and to derive situation specifications using a logical disjunction between these conjuncted predicates. To represent complicated knowledge (especially temporal logic [40]), the capability of expressing logic in context lattices is far insufficient. In the future, we will extend API with logical programming and temporal reasoning [41] to support representing and evaluating more complicated logical expressions.

8.2. Capability in Inference and Uncertainty Resolving

Figure 10 shows that the context lattice performs better in the TK26M data set than in the PlaceLab data set. By examining and comparing the two data sets, we find that (1) the activities recorded in PlaceLab diary cover much more activities and some of them barely have an identifiable pattern (e.g., "using phone" given that there is no effective sensors on the phone), while the activities recorded in the TK26M diary are mostly identifiable immediately from the sensors; and (2) the PlaceLab data set involves external noise in the data set, since it has involved two participants and most sensors are not person-specific.

These two causes are critical for context lattices, since the way that context lattices are created is to combine the *relevant* sensor data. This only captures a local snapshot of the reality, while other techniques like the decision tree and Naive Bayes start with all the sensor data and capture a universal snapshot of the reality. This universal combination might be useful in recognising activities from sensor data with more uncertainty, if the other local views can serve as an evidence supporting or complementing the current local

snapshot. However, the scalability issue of a context lattice keeps it from building a universal combination, which makes it perform not as well as the other two techniques.

When activities have identifiable patterns from available sensors, context lattices have presented better accuracy than the other techniques. The reason is that context lattices are defined and constructed using expert knowledge, so the learning process will be responsible for "adding" relationships between sensor data and situations. To resolve uncertainty of sensor data to a certain degree, developers can simply introduce well-known domain knowledge into a context lattice, which will potentially increase the accuracy. In contrast to the machine learning techniques, context lattices can effectively integrate domain knowledge and sensor data, which results in low demand on the amount of training data and high accuracy in inferring.

When it comes to resolving uncertainty, currently we used the naive training techniques, as no intelligent techniques or major result in lattice theory have been applied. This makes a context lattice less resistant to noise of sensor data, which result in worse performance in recognising situations than classic machine learning techniques, even though they do not need extra knowledge from developers while the context lattice is built with expert knowledge.

Some work has been done in uncertainty reasoning by applying possibility theory to lattice-valued logic [42,43]. We are investigating different uncertainty resolving techniques in the context lattice, such as Bayesian network [4] and Fuzzy logic [36]. Since the preliminary results are promising, we will work on configuring the context lattice with these different techniques and conduct a more comprehensive evaluation to find out which technique best suits which type of uncertainty.

8.3. Knowledge Engineering Effort

The strength of a context lattice is to allow developers to express and use domain knowledge. Using the basic structural feature of a lattice, developers can represent the semantic relationships between context predicates and restrictions of situations on context predicates (i.e., what situations cannot occur on context predicates) can be represented.

When there are a large number of preliminary nodes (e.g., over hundreds), developers may need to define the combination mechanism between types of context predicates so as to reduce the complexity of the generation process. We have demonstrated how to group and combine the nodes on the PlaceLab data set, which is straightforward and manageable. To our best knowledge, this data set is the most complicated among the existing data sets, which covers the largest number of sensors. Therefore, we expect the process will be much simpler on other smaller data sets if applicable.

The context lattice allows developers to incorporate domain knowledge to guide and restrict the learning process. The result in Figure 11 has demonstrated that the accuracy of inference can be greatly improved by incorporating a small amount of domain knowledge, which is intuitive and easy to specify. We agree that the added knowledge is to solve particular uncertainty issues in particular environments, but there is no huge amount of knowledge engineering effort involved in the process. The addition of the domain knowledge is necessary only when there exists external noise in the data set and the noise can be simply dealt with by constraining the relation between certain sensors and activities. When the data set is relatively clean (e.g., the TK26M data set), the developers do not need to add any domain knowledge.

8.4. Scalability

A context lattice can scale well with the addition of new situations, since it does not demand much training data or effort in restructuring. To contain a new situation, developers only need the training data about the new situation and the sensor data during its occurrence. A context lattice will "label" it on the nodes that map to these sensor data, without "reshaping" the structure of a context lattice.

When an environment is introduced with a new sensor, a few primitive nodes will be created based on the values produced from this sensor. These new nodes will be combined with the existing nodes in a context lattice, while its original structure will not be changed. Developers will need to train the new context lattice with the sensor data (including the existing sensors and new sensors) and newly recorded diary. However, the more sensors involved, the more context predicates defined, which will raise the scalability issue. To address this problem, this paper demonstrates a means to decrease the complexity by grouping the predicates and specifying their combination policy. This means introduces subjectivity, which might affect the performance of situation inference as discussed in Section 8.2.

It is suggested that if an environment involves a limited number of sensors (such as the TK26M data set),

developers should define context predicates according to sensor inputs and let context lattices to generate all the combinations; if an environment involves a number of different types of sensors (such as the Placelab data set), developers should define context predicates in a semantic way and add more knowledge in specifying the combination policy.

9. Conclusion and Future Work

This paper describes a new data structure, context lattices, to recognise activities in a smart home environment. The novelty of this structure is its ability to represent and use low-level semantics on domain knowledge and sensor data and to derive high-level semantics on human activities. The low-level semantics helps to uncover correlations between sensor data and activities while the high-level semantics provide guidance on situation-aware application design. The evaluation results in Section 6 have shown that a context lattice can recognise activities at a comparable accuracy to traditional machine learning techniques while consuming less training data.

As presented in Section 8, our future work will focus on extending the theoretical model with the possibility theory such that context lattices will perform better in dealing with noisy data. A context lattice suffers a severe scalability issue, which keeps it from automatically generating all the combinations of logical expressions that involve the logical disjunction and negation. In the future, we will look into the ways to resolve the scalability issue such as [44,45].

The current API needs to be incorporated with logical programming to support representing and evaluating more complex logical formulas. In addition, the API should be made more friendly to end developers so as to facilitate the process of defining context predicates and even visualise the lattice.

References

[1] Diane J. Cook, Juan C. Augusto, and Vikramaditya R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298, August 2009.

[2] Hideyuki Nakashima, Hamid Aghajan, and Juan Carlos Augusto, editors. *Handbook of Ambient Intelligence and Smart Environments*. Springer Verlag, 2009.

[3] Graham Thomson, Sotirios Terzis, and Paddy Nixon. Situation determination with reusable situation specifications. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 620–623, 2006.

[4] Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon. Representing and manipulating situation hierarchies using situation lattices. *Revue d'Intelligence Artificielle*, 22(5):647–667, 2008.

[5] Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon. Using situation lattices in sensor analysis. In *Proceedings of PerCom 2009*, pages 1–11, March 2009.

[6] Beth Logan, Jennifer Healey, Matthai Philipose, Emmanuel Munguia Tapia, and Stephen S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of Ubicomp 2007*, pages 483–500, Innsbruck, Austria, September 2007.

[7] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *Proceedings of UbiComp '08*, pages 1–9, New York, NY, USA, September 2008. ACM.

[8] H. A. Kautz. *A formal theory of plan recognition*. PhD thesis, University of Rochester, Rochester, NY, USA, 1987.

[9] Seng W. Loke. Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. *Knowledge Engineering Review*, 19(3):213–233, 2004.

[10] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: models and approach. *Pervasive and Mobile Computing*, 2(1):37–64, February 2006.

[11] Stephen S. Yau and Junwei Liu. Hierarchical situation modeling and reasoning for pervasive computing. In *Proceedings of SEUS-WCCIA'06*, pages 5–10, 2006.

[12] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[13] Nuria Oliver, Ashutosh Garg, and Eric Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision Image Understanding*, 96(2):163–180, 2004.

[14] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. In *Proceedings of the second International Conference on Pervasive Computing*, pages 1–17, Vienna, Austria, April 2004.

[15] Tao Gu, Xiao Hang Wang, Hung Keng Pung, and Da Qing Zhang. An ontology-based context model in intelligent environments. In *Proceedings of CNDS 2004*, pages 270–275, January 2004.

[16] Christian Wojek, Kai Nickel, and Rainer Stiefelhagen. Activity recognition and room-level tracking in an office environment. In *Proceedings of MFI '06*, pages 25–30, September 2006.

[17] Md. Kamrul Hasan, Husne Ara Rubaiyeat, Yong-Koo Lee, and Sungyoung Lee. A hmm for activity recognition. In *Proceedings of ICACT 2008*, volume 1, pages 843–846, 2008.

[18] Anind K. Dey. Understanding and using context. *Personal Ubiquitous Computing*, 5(1):4–7, 2001.

[19] Xiao Hang Wang, Tao Gu, Da Qing Zhang, and Hung Keng Pung. Ontology Based Context Modeling and Reasoning using OWL. In *Workshop on Context Modeling and Reasoning(CoMoRea 2004)*, pages 18–22, March 2004.

[20] Patricia Dockhorn Costa, Giancarlo Guizzardi, Joao Paulo A. Almeida, Luis Ferreira Pires, and Marten van Sinderen. Situations in conceptual modeling of context. In *Proceedings of EDOCW'06*, pages 6–16, Hong Kong, China, October 2006.

[21] Stephen S. Yau, Dazhi Huang, Haishan Gong, and Yisheng Yao. Support for situation awareness in trustworthy ubiquitous computing application software. *Software: Practice and Experience*, 36(9):893–921, 2006.

[22] Hung H. Bui, Svetha Venkatesh, and Geoff West. Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. *Hidden Markov models: applications in computer vision*, pages 177–196, 2002.

[23] T. van Kasteren and B. Krose. Bayesian activity recognition in residence for elders. In *Proceedings of the third IET International Conference on Intelligent Environments*, pages 209–212, September 2007.

[24] Emmanuel M. Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing*, pages 158–175, 2004.

[25] B. Clarkson and A. Pentland. Unsupervised clustering of ambulatory audio and video. In *Proceedings of ICASSP '99*, pages 3037–3040, 1999.

[26] Thi V. Duong, Hung H. Bui, Dinh Q. Phung, and Svetha Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *Proceedings of CVPR'05 - Volume 1*, pages 838–845, 2005.

[27] D. Minnen, T. Starner, J.A. Ward, P. Lukowicz, and G. Troster. Recognizing and discovering human actions from on-body sensor data. In *Proceedings of ICME 2005*, pages 1545–1548, July 2005.

[28] Lin Liao, Dieter Fox, and Henry Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *International Journal on Robotics Research*, 26(1):119–134, 2007.

[29] Kai Kunze and Paul Lukowicz. Dealing with sensor displacement in motion-based onbody activity recognition systems. In *Proceedings of UbiComp '08*, pages 20–29. ACM, 2008.

[30] Joseph Modayil, Tongxin Bai, and Henry Kautz. Improving the recognition of interleaved activities. In *Proceedings of UbiComp '08*, pages 40–43, 2008.

[31] Garrett Birkhoff. *Lattice Theory*. Providence, R.I. : American Mathematical Society, 3rd edition, 1967.

[32] Juan Ye and Simon Dobson. Human-behaviour study with situation lattices. In *Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics*, pages 343–348, October 2009.

[33] Peter Burmeister and Richard Holzer. Treating incomplete knowledge in formal concept analysis. In *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, pages 114–126. Springer, 2005.

[34] Bernhard Ganter and Rudolf Wille. Applied lattice theory: Formal concept analysis. Technical report, Technical University of Dresden, Germany, 1997.

[35] Uta Priss. Lattice-based information retrieval. *Knowledge Organisation*, 27(3):132–142, 2000.

[36] Juan Ye, Susan McKeever, Lorcan Coyle, Steve Neely, and Simon Dobson. Resolving uncertainty in context integration and abstraction. In *Proceedings of the international conference on Pervasive Services*, pages 131–140, New York, NY, USA, July 2008. ACM.

[37] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

[38] Lorcan Coyle, Juan Ye, Susan McKeever, Stephen Knox, Matthew Stabeler, Simon Dobson, and Paddy Nixon. Gathering datasets for activity identification. In *Proceedings of the workshop on developing Shared Home Behaviour Datasets to Advance HCI and Uniquitous Computing Research at CHI 2009*, April 2009.

[39] Judy Kay, Bob Kummerfeld, and Piers Lauder. Managing private user models and shared personas. In *Workshop on User Modelling for Ubiquitous Computing, coexisted with 9th International Conference on User Modeling*, 2003.

[40] James Allen and George Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

[41] Juan Carlos Augusto and Chris Nugent. A new architecture for smart homes based on ADB and temporal reasoning. In *Proceedings of 2nd International Conference On Smart homes and health Telematic*, pages 106–113. IOS Press, 2004.

[42] Shuwei Chen, Yang Xu, and Jun Ma. A linguistic truth-valued uncertainty reasoning model based on lattice-valued logic. *Fuzzy Systems and Knowledge Discovery*, 3613/2005:276–284, 2005.

[43] Didier Dubois, Florence Dupin de Saint-Cyr, and Henri Prade. A possibility-theoretic view of formal concept analysis. *Fundam. Inf.*, 75(1-4):195–213, 2007.

[44] Liming Chen and Christopher Nugent. Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 5:410–430, 2009.

[45] Daniele Riboni and Claudio Bettini. Context-aware activity recognition through a combination of ontological and statistical reasoning. In *Proceedings of the international conference on Ubiquitous Intelligence and Computing*, pages 39–53. Springer Berlin/Heidelberg, 2009.