A smart brain: an intelligent context inference engine for context-aware middleware

Tao Xu*

School of Software and Microelectronics,
Northwestern Polytechnical University,
127 West Youyi Road,
Xi'an Shaanxi, 710072, China
Email: daridxu@gmail.com

Yun Zhou*

School of Education,
Shaanxi Normal University,
199, South Chang'an Road,
Xi'an Shaanxi, 710062, China
Email: chouyun920@gmail.com
*Corresponding authors

Bertrand David and René Chalon

Université de Lyon, CNRS,
Ecole Centrale de Lyon, LIRIS, UMR5205,
Ecully, 69134, France
Email: bertrand.david@ec-lyon.fr
Email: rene.chalon@ec-lyon.fr

Abstract: Recently, 'activity' context draws increased attention from researchers in context awareness. Existing context-aware middleware usually employ the rule-based method to, which is easy to build and also intuitive to work with. However, this method is fragile, not flexible enough, and is inadequate to support diverse types of tasks. In this paper, we surveyed the related literature in premier conferences over the past decade, reviewed the main activity context recognition methods, and summarised their three main facets: basic activity inference, dynamic activity analysis, and future activity recommendation. Based on our previous work, we then proposed an intelligent inference engine for our context-aware middleware. Besides satisfying requirements for checking context consistency, our inference engine integrates the three methods for activity context recognition to provide a solution for all facets of activity context recognition based on our context-aware middleware.

Keywords: context awareness; context reasoning; activity recognition; middleware; sensor networks.

Reference to this paper should be made as follows: Xu, T., Zhou, Y., David, B. and Chalon, R. (2016) 'A smart brain: an intelligent context inference engine for context-aware middleware', *Int. J. Sensor Networks*, Vol. 22, No. 3, pp.145–157.

Biographical notes: Tao Xu is currently an Assistant Professor in Northwestern Polytechnical University. He got his PhD in Ecole Centrale de Lyon in 2013. He received his BE and ME in Automation and Software Engineering at Xi'an Jiaotong University in 2006 and 2009, respectively. His research activities focus on ubiquitous computing, context awareness, machine learning and middleware.

Yun Zhou received her PhD from Ecole Centrale de Lyon in 2012. She received her BE from Beijing Normal University and ME in Educational Technology from Beijing Jiaotong University in 2006 and 2009, respectively. Her research interests focus on user interface design, wearable computing and evaluation.

Bertrand David is a Professor in Computer Science at Mathematics and Computer Science Department of Ecole Centrale de Lyon and a Researcher in LIRIS Lab (Laboratoire d'InfoRmatique en Image et Systèmes d'information), SILEX team. His research interests include human-computer interaction, ambient intelligence and mobile learning.

René Chalon is an Associate Professor in Computer Science at Mathematics and Computer Science Department of Ecole Centrale de Lyon. He has been a Researcher in LIRIS lab (Laboratoire d'InfoRmatique en Image et Systèmes d'information), SILEX team, since July 2011. His research interests include human-computer interaction and ambient intelligence.

This paper is a revised and expanded version of a paper entitled 'Supporting activity context recognition in context-aware middleware' presented at *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI'13)*, Bellevue, Washington, USA, 14–18 July, 2013.

1 Introduction

Almost 20 years earlier, Marc Weiser viewed the prospect of computers in the 21st century and proposed the pioneering notion of ubiquitous computing. Most of his visions have already been realised in the past two decades. Furthermore, one of his primary visions, namely that systems could adapt their functionality to a user's activity and situation in the environment, has led to a new field: context awareness (Lukowicz et al., 2012). Today, context awareness, as one of the central themes in the Smart Space (also described as ambient intelligence), is attracting increasing attention from researchers. Context-aware systems are concerned with acquisition of context (e.g., using sensors to perceive a situation), abstraction and understanding of context (e.g., matching a perceived sensory stimulus to a context) and application behaviours based on the recognised context (e.g., triggering actions based on the context) (Schmidt et al., 1998). To achieve this goal, we need to incorporate the sensors, actuators, communication objects and computing devices required for the system. While low-level mechanisms and drivers are necessary, they must be encapsulated into the more common and higher-level view designed to collect, treat and disseminate information appropriately among these components. Namely, the system should take into account changes in context and propagate appropriate decisions (Xu et al., 2011). Context-aware middleware provides a feasible solution, which meets the aforementioned requirement.

The holy grail of context awareness is to divine or understand human intent (Krumm, 2009). An intelligent system should be able to provide natural interaction between users and the physical environment. Also, a 'smart' context-aware middleware is required to support development of the smart space. Context inference plays the role of the brain in context-aware middleware. Smart space needs a smart brain, namely an intelligent context inference engine. Context inference provides two main services: the first is to check context consistency, whereas the second is to determine or infer the user's situation. Once the user's situation has been inferred, the application can then take appropriate action (Krumm, 2009). In recent years, most context-aware middleware have adopted web ontology language (OWL) to build the ontology-based model, which provides an efficient solution to check context consistency. Consequently, research now focuses on the second service: infer the user's situation (commonly referred to as activity). The existing context-aware middleware usually employs the rule-based method to infer user's activity, such as the study conducted by Wang et al. (2004) concerning leveraging rules based on first-order logic. To better recognise the user's activity, a large number of artificial intelligence methods are explored and used in context-aware computing. Korpipaa et al. (2003) use a naïve Bayes classifier to distinguish higher-level contexts from lower-level contexts. The HMM is employed to recognise activities in a smart room (van Kasteren et al., 2008). Pollack et al. (2003) use a decision tree to make decisions about whether and when it is most appropriate to issue reminders for prescribed activities. However, these approaches only provide a possibility to solve the problem partly based on activity context recognition. None of them has considered integration of the approach into the context-aware system, which encourages collaborative work with other parts of the system.

In this paper, we focus on designing a context inference engine, which supports activity context recognition and improves context-aware middleware intelligence. We first present our context-aware middleware. We then review all the methods relating to activity context recognition that have been published in the three premier conferences in the past decade. To go one step further, we conclude that activity context recognition consists of three facets: basic activity inference, dynamic activity analysis and future activity recommendation. Integrating these three most salient methods of activity context recognition used in context-aware applications via strategy patterns, we propose an intelligent inference engine based on our context-aware middleware. An invoking mechanism for the inference engine is designed to provide services and deal with tasks. This intelligent inference engine not only provides a solution for meeting current requirements in the field of activity context recognition, but can also be upgraded to meet future demands.

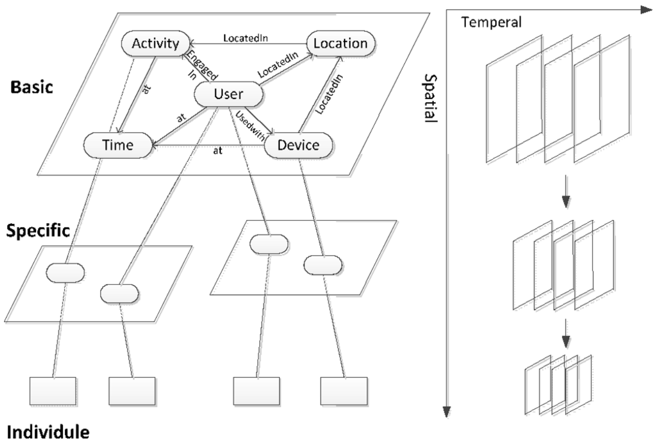
The remainder of the paper is organised as follows: First, we describe our spatial-temporal ontology-based model. Second, we present the architecture of context-aware middleware working for the smart space. Third, we conclude as to the main problems in activity context recognition and summarise the main approaches already existing. Following these discussions, we propose an intelligent inference engine for our context-aware middleware. Furthermore, two scenarios are investigated to explain and verify the conceptual work. Finally, we conclude our work, discuss the pros and cons of our intelligent inference engine and envisage our future study prospects.

2 The spatial-temporal ontology-based model

Before detailing context-aware middleware, we will describe our spatial-temporal ontology-based model. A context model, as a fundamental part of the context-aware system, aims at defining and storing context data in a machine-readable form Baldauf et al. (2007). Developing a flexible and useable context model that covers a wide range of possible contexts is a challenging task (Stojanovic, 2009). We adopt the ontology-based context model to construct our context-aware middleware. In the field of context awareness, ontology is a reference model for components and behaviours of context (Wang et al., 2004). The ontology-based model has a large number of good features for developing the context-aware system, such as knowledge sharing, knowledge reuse and logic inference. In particular, logic inference enables the application to use directly deduced high-level context information.

From the spatial dimension, we employ a hierarchical structure to describe the user's situation and circumstance based on OWL, which is an ontology markup language adopted by W3C as standard for semantic web. The structure is shown in Figure 1. The basic model defines generic conceptions and relationships in the Smart Space, which come up with a basic context structure. It has five interrelated basic classes: user, location, time, activity and device, as well as eight properties (relationships) among classes, which represent who, where, when, what and why. Basic context-aware ontology can be completed and upgraded by more precise information related to a particular application or application area. It is considered as the specific model, which can be reused and shared in different domains.

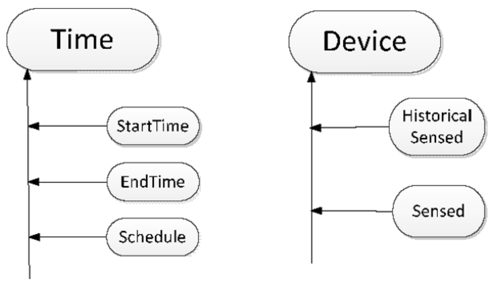
Figure 1 The spatial-temporal ontology-based model



The context temporal model is an attempt to organise context according to the temporal dimension. It reflects two primary classes: time and device shown in Figure 2. The time class provides three subclasses to describe the temporal relationship between user and activity. The device class uses the subclass 'historical sensed' to record historical context data, which can be used to analyse users' activities and provide predictions and recommendations.

The basic context-aware model is developed by context-aware middleware designers, whereas the specific model is developed by context-aware application developers.

Figure 2 The temporal model class



3 Context-aware middleware

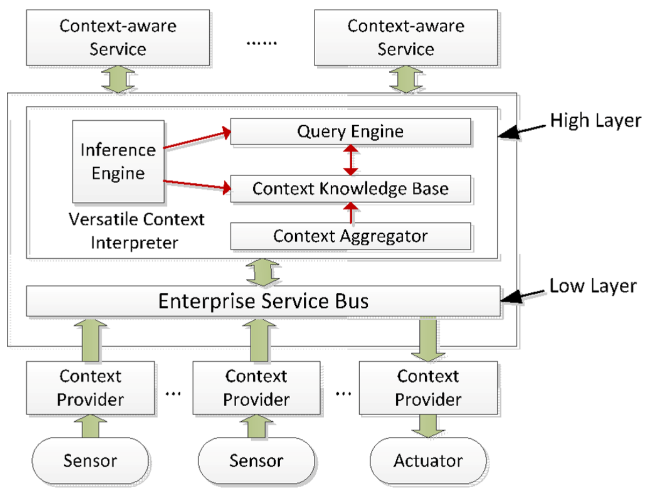
Context-aware applications are becoming increasingly salient, and are also used prevalently in the areas of wearable computing, intelligent environments, context-sensitive interfaces, etc. (Krumm, 2009). A generally accepted definition of context is given by Dey and Abowd (Abowd et al., 1999): "Any information that can be used to characterise the situation of an entity. An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves". The context-aware system is defined as the system that uses the context to provide the relevant information and services to the user, where the relevancy depends on the user's task.

Development of context-aware applications is inherently complex. These applications adapt to the context information: physical context, computational context and user context/tasks (Ballendat et al., 2010). It requires low-level mechanisms to integrate all sensors, actuators, communication objects and computing devices into the system. Then, we propose to create an application-independent common high-level contextualisation making it possible to collect, process, interpret and propagate information with the context model and reasoning mechanisms.

To implement this more in-depth approach of context-aware services in the smart space, we have designed a context-aware middleware based on mobility, contextualisation and cooperation (MOCOCO), organised in two layers as shown in Figure 3.

The low layer is a sort of Enterprise Service Bus. It allows services to be easily plugged in and out of the network without any impact on other components and without the need to restart the system or even stop running applications. In our paper, the context provider is the service used to obtain context from sensors, the web or other sources, as well as dispatching commands to actuators. The low layer provides a unified standard interface to achieve the core functions of service interaction: service registry, dynamic service discovery and service consumption. It also integrates interaction devices and a set of application program interface (API) for different interaction modalities to support development of interaction approaches.

Figure 3 Context-aware middleware architecture (see online version for colours)



The versatile context interpreter (VCI) is a high layer of context-aware middleware, made up of four parts: context aggregator, inference engine, context knowledge base and query engine. It leverages the low layer's basic services results to deliver and manage context-aware views and interpretations to deliver high-level information to the application.

- *The context aggregator* is responsible for working with basic contextual data collected by the low layer.
- *The context knowledge base* provides persistent storage for context through the use of relational databases, as well as supplying a set of library procedures for other components to query and modify context knowledge.
- *The context query engine* offers two main services: The first is to handle queries from the application. It supports SPARQL, which is an resource description framework (RDF) query language, able to retrieve and manipulate data stored in OWL. The second is to invoke the context inference engine. When the application requests high-level context, it will invoke the context inference engine to generate the inferred context.
- *The intelligent inference engine* is the central and intellectual component of our context-aware middleware. Its details will be presented in the next section.

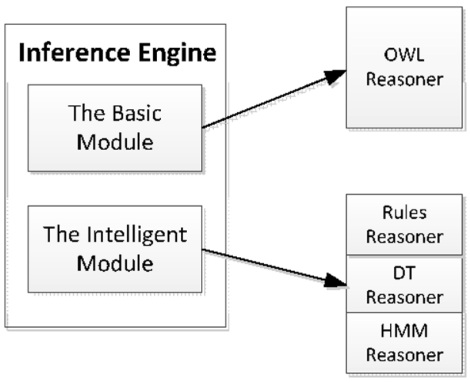
4 The intelligent inference engine

Context inference, often described as context reasoning, refers to inferring further the implicit context from the explicit context. As we mentioned earlier, its main task focuses on two main aspects: one task is to check and solve inconsistencies in context data, whereas the other is to deduce high-level context information from low-level context data. The high-level context is commonly considered as the activity.

With respect to context-aware computing, researchers focus on who's, where's, when's and what's (namely, what activities are occurring) of entities and use this information to determine the reason why a situation is occurring (Krumm, 2009). Research work on context awareness is simplified to achieve these five '*w*'. Among these '*w*', the 'why' is the destination, whereas the other four '*w*' (who, where, when and what) represent four fundamental contexts: user, location, time and activity, respectively. With the rapid development of technology and widespread use of smart mobile devices and sensors, the first three fundamental contexts (user, location and time) are now relatively easily captured directly from context sources. However, recognising user's activities continues to be difficult.

To ensure better understanding of human intent, we propose an intelligent inference engine, consisting of a basic module and an intelligent module as shown in Figure 4. Besides satisfying the basic requirements, it can provide appropriate methods for different facets of activity context recognition, respectively.

Figure 4 The inference engine structure



The basic module's main task is to check context consistency. We adopt the ontology-based context model to construct our context-aware system. It exhibits good features for developing the context-aware system, such as knowledge sharing, knowledge reuse and logic inference. Also, it paves the way for the basic inference module. OWL has a built-in reasoner based on description logic. This reasoner can fulfil the essence of logical requirements, which comprises concept satisfiability, class subsumption, class consistency and instance checking. Table 1 shows a partial basic reasoning rule.

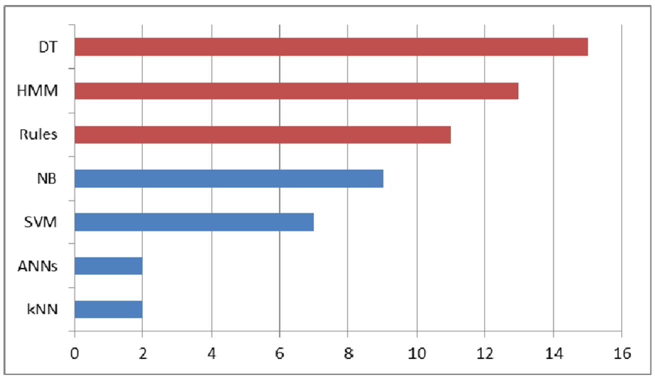
Table 1 The partial basic reasoning rule for OWL

Transitive property	P(x, y) and P(y, z) implies P(x,z)
Symmetric property	P(x, y) iff P(y, x)
Functional property	P(x, y) and P(x, z) implies y = z
inverseOf	P_1 (x, y) iff P_2(y, x)
Inverse functional property	P(y, x) and P(z, x) implies y = z

The intelligent module is responsible for deducing high-level context. In our paper, high-level context mainly refers to activity. Our aim is to provide a solution supporting activity context recognition.

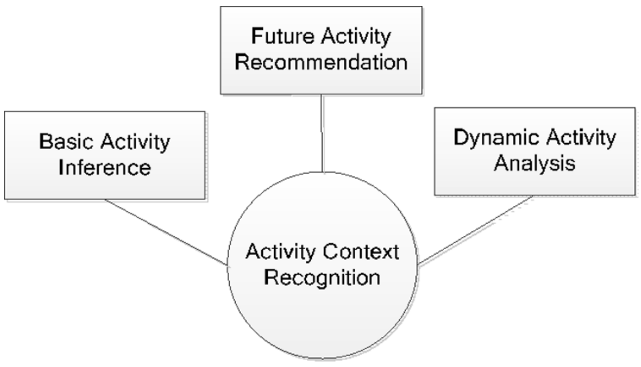
Activity context recognition aims at inferring a user's behaviour from observations such as sensor data (Hu et al., 2011), and serves a variety of applications including medical care (Pollack et al., 2003), logistics service (Lin, 2006), robot soccer (Vail et al., 2007), plan recognition (Geib et al., 2008), etc. Many researchers focus on finding the most efficient way to recognise user's activities. On the basis of Lim and Dey's (2010) work, we reviewed literature from three top-tier conferences on context-aware computing in recent years, including the ACM Conference on Human Factors in Computing Systems (CHI) from 2003 to 2012, the ACM International Conference on Ubiquitous Computing (Ubicomp) from 2004 to 2012 and the International Conference on Pervasive Computing (Pervasive) from 2004 to 2012. Fifty-nine papers relate to user activity recognition, involving seven algorithms: rules, decision tree (DT), Naïve Bayes (NB), HMM, support vector machine (SVM), k-nearest neighbour algorithm (kNN) and artificial neural networks (ANNs). The detailed statistical results are shown in Figure 5.

Figure 5 The statistical results of the activity recognition algorithm (see online version for colours)



We conclude on the research conducted into recognising user's activity in three facets shown in Figure 6: basic activity inference, future activity recommendation and dynamic activity analysis.

Figure 6 The three main facets of activity context recognition


Future Activity Recommendation
Basic Activity Inference
Dynamic Activity Analysis
Activity Context Recognition

- *Basic activity inference* focuses on recognising the basic types of activity, such as working, sleeping and shopping. User's activity can be deduced directly based on low-level context obtained from diverse context sources containing physical context sensors and virtual context source (web service). The method based on rules is used to infer the basic activity.
- *Future activity recommendation* usually refers to recommending or predicting user's future activity or choices. To achieve this, besides taking into consideration the user's current situation (context), the context of the user's previous similar behaviours (training data) must be analysed. The decision tree can ideally cope with this problem, and can be considered as a type of context-aware recommendation.
- *Dynamic activity analysis* is responsible for analysing fine-grained activity compared with basic activity inference. There are two main solutions: the first employs the state-space model, which partitions the activity into state sequences and infers the type of activity (state sequence) based on the probability of an observed sequence such as Bayesian networks and HMM. The second solution relies on pattern recognition techniques, which extract patterns from activities and infer the type of activity based on different activity patterns, such as SVM and ANN. As it is similar to future activity recommendations, it also requires learning of process parameters from the previous activity information.

We can observe that the three most popular methods are rules, DT and HMM. Each method has its own advantages and characteristics. We have analysed them (detailed information will be stated in the following sections) and found that all three methods can be leveraged to deal with these three facets, respectively. Moreover, these three methods mostly represent the mainstream solution on their actual facet. We integrate the three methods, namely rules, DT and HMM, into our context-aware middleware as our research focus.

4.1 Rules

'Rules' is an early attempt to infer user's activity in context-aware computing. 'Rules' refers to the method using a set of if–then rules to infer user's activity based on first-order logic: if the devices sense a particular situation, then it can deduce the user's activity. 'Rules' is based on general features of activity. It should be possible to obtain these chosen features of activity, namely low-level context, by physical sensors or other context sources. We employ a simple example as shown in Figure 7 to interpret it.

'Rules' has a wide range of adaptabilities, which is usually designed to provide inference for almost all users. Furthermore, it is relatively intuitive and easy to work with. However, rules are rigid, meaning that they are also brittle

(Krumm, 2009): even trifling exceptions will generate some errors. Nevertheless, since this method is easy to implement in context-aware applications, it continues to be an effective and widely used method to infer user's activity. We leverage it to work on basic activity inference.

Besides rules, the common alternative approaches involve artificial intelligence algorithms. DT and HMM both belong to this category, and each has its own characters to focus on different facets.

Figure 7 This is an example of rules

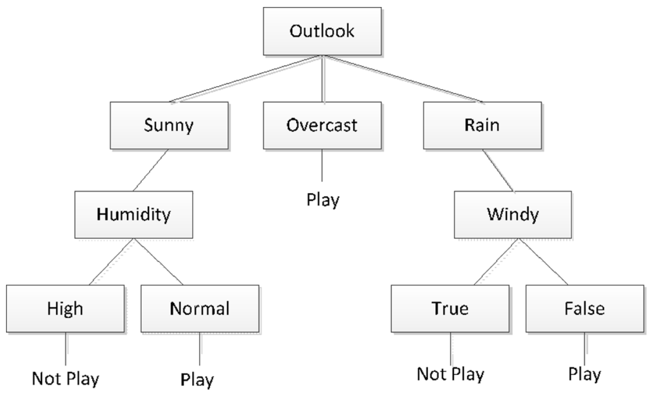
```
@prefix ex:<http://liris.cnrs.fr/smartspace.owl#>
[Working:
        (?event ex:has?time) greaterThan (?time,9)
        (?person ex:hasLocationIn ex:lab)
        (?light ex:hasStatus ex:ON)
        (?door ex:hasStatus ex:ON)
 ->
        (?person ex:isDoing ex:work)]
```

4.2 Decision Tree (DT)

The decision tree is a classic algorithm in the field of artificial intelligence shown in Figure 8. It is a predictive model that maps observations on an item to conclude on the item's target value. Decision trees are learned by recursively partitioning training data into subgroups until those subgroups contain only instances of a single class. Processing for partitioning data runs on the values of item attributes. The choice of the item attribute on which to operate the partition is generally made according to the entropy criterion and the information gain. The entropy of *S* can be described in function (1), which is a measurement of the expected encoding length in bits.

$$\text{Entropy(S)} = \sum -p(\text{s}) \log_2 p(\text{s}) \tag{1}$$

Figure 8 The classic example of the decision tree



The information gain is the expected reduction in entropy caused by partitioning the examples according to this attribute. The information gain, Gain(*S*, *A*) of an attribute *A*, relative to a collection of examples *S*, is defined as:

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Value(A)}} \frac{|S_v|}{|S|} \text{Entropy}(S_v) \tag{2}$$

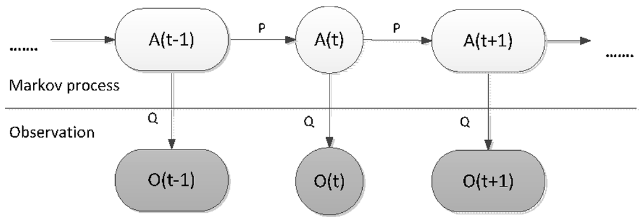
where Values(*A*) is the set of all possible values for attribute *A* and S_v is the subset of *S* for which attribute *A* has the value *v*. The information gain is used to select the best attribute at each step for growing the tree. More clearly, the classic example is shown here.

The decision tree is simple to understand and interpret. Users are able to understand decision tree models following a brief explanation. Decision trees are popular for their simplicity of use, interpretability and good runtime performance. They are commonly adopted in content-based recommender systems (Pazzani et al., 1996; Pollack et al., 2003), which can be employed to give a recommendation for user's activity.

4.3 Hidden Markov model (HMM)

The HMM is one of the most accepted algorithms in temporal recognition tasks, including speech, gesture, activity, etc. The HMM is a statistical Markov model, which can recover a data sequence that is not immediately observable. In human activity recognition, complex activities have a temporal structure. The time series data obtained by sensors is divided into time slices of constant length, where each slice is labelled with a state of activity. A generic HMM for activity is illustrated in Figure 9. The shaded nodes (*A*) represent observable variables (data from sensors), whereas the white nodes (*Q*) represent hidden variables (state of activity). *t* in these two functions represents the time of state, *P* is the state of transition probabilities and *Q* is the observation probability.

Figure 9 The simple example of the HMM for activity recognition



The HMM can learn the parameters '*P*, *Q*' of the probabilistic model from the training data. Inference, which best labelled sequence explaining the new coming data from the sensors, is depended on calculating a maximum of the conditional probability $P(a_1, a_2, a_3 ... | o_1, o_2, o_3 ...)$:

$$a_1, a_2, a_3 ... = \underset{\text{all } a_1, a_2, a_3 ...}{ArgMax}\, P(a_1, a_2, a_3 ... | o_1, o_2, o_3 ...) \tag{3}$$

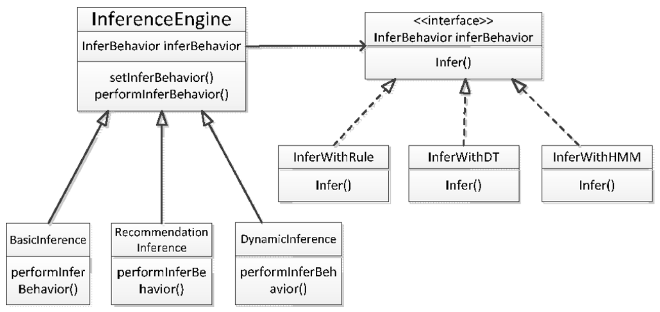
where a_i presents a state of activity and o_i refers to observable variables from the sensors. HMM is rapidly gaining ground in dynamic activity analysis.

4.4 Organisation of three algorithms

Our context-aware middleware enables application designers to concentrate on the development of application logic. The intelligent inference engine takes into account two aspects of design: expandability and scalability. Rules, DT and HMM play important roles in basic activity inference, future activity recommendation and dynamic activity analysis, respectively. The three algorithms are related to various implementation methods. We adopt the strategy pattern to organise the three algorithms. This encapsulates the algorithm into separate classes, which enable the context-aware application developer to vary the algorithm independently from the context and to plug in a new one at runtime. The strategy pattern offers an alternative to conditional statements for selecting desired behaviour, which makes the three algorithms interchangeable. It gives this module flexibility, so that context-aware application developers can alter and extend the module.

We define a universal interface: 'InferBehaviour'. Context-aware application developers can add other algorithms without affecting the original ones, as long as the algorithms employ this interface. Moreover, if context-aware application developers want to adopt different inference methods to deal with different situations, they just invoke the function with the same name based on this interface, and do not need to use the actual algorithm. The partial UML model is shown in Figure 10.

Figure 10 The partial UML of the inference engine



4.5 *The invoking mechanism*

The invoking mechanism refers to how and when the context-aware middleware runs the inference engine. According to different reasoner tasks in the intelligent inference engine, we present different methods for the reasoners, respectively.

In the basic module, the OWL reasoner's main task is to check context consistency. It is invoked when the new context model is added, such as adding a new specific model. In the intelligent module, owing to the different methods of activity context recognition, the invoking mechanism is more complex than the basic module.

Concerning the rules reasoner, we design a query invoking mechanism that enables the rules reasoner to work when receiving the query from an external service, such as context-aware applications. There is a very interesting phenomenon, i.e., the property exists in almost all the query statements: for example: '?s ex:hasLocated ex:Room3'. Furthermore, the property is like a linking point of the objects, the amount of which is much less than the object itself. So, these properties are defined as the keywords, which are used to trigger the specific users' rules set in the rules reasoner. When a query statement arrives, the context-aware middleware reacts: it parses the query statement, then tries to match keywords. If it succeeds, the basic inference engine is invoked, while, on the contrary, if it fails, it searches for the context database directly. The entire flow is shown in Figure 11.

With respect to the DT reasoner, we present a schedule invoking mechanism, which invokes the DT reasoner based on the user's schedule. The DT reasoner is in charge of future activity recognition, which works with the user's schedule. When the system finds that the user is not in the place where he/she is scheduled, the system reminds him/her and gives a suggested choice based on the user's current location. Figure 12 explains the flow of schedule of the trigger mechanism.

Figure 11 The flow chart of the query invoking mechanism

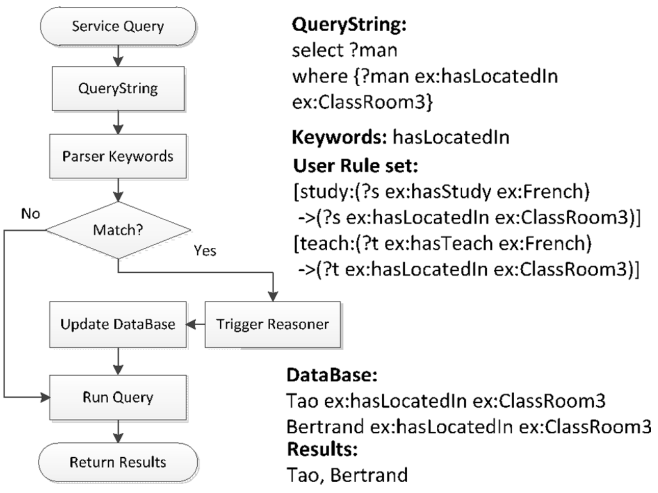
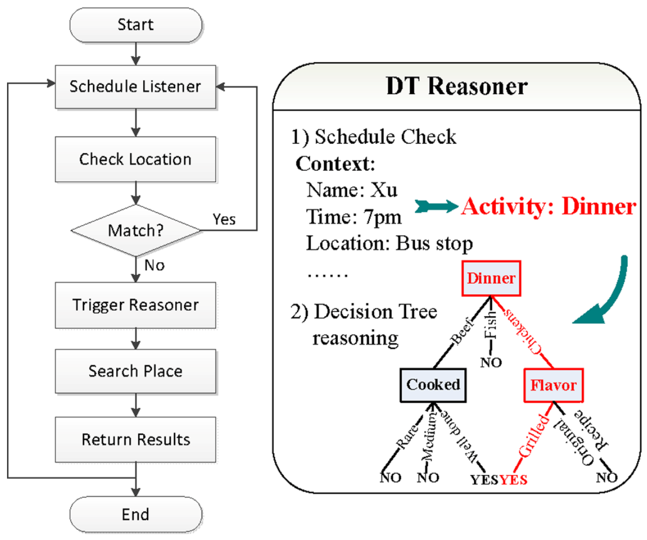

Service Query
QueryString
Parser Keywords
Match?
No
Yes
Update DataBase
Trigger Reasoner
Run Query
Return Results
QueryString:
select ?man
where {?man ex:hasLocatedIn
ex:ClassRoom3}
Keywords: hasLocatedIn
User Rule set:
[study:(?s ex:hasStudy ex:French)
->(?s ex:hasLocatedIn ex:ClassRoom3)]
[teach:(?t ex:hasTeach ex:French)
->(?t ex:hasLocatedIn ex:ClassRoom3)]
DataBase:
Tao ex:hasLocatedIn ex:ClassRoom3
Bertrand ex:hasLocatedIn ex:ClassRoom3
Results:
Tao, Bertrand

Figure 12 The flow chart of the schedule invoking mechanism (see online version for colours)



We enable the HMM reasoner to always work when the middleware system is running, owing to the requirements of the reasoner's algorithm and task. This provides the activity sequence inference based on the sequence sensor data, and thus requires time series data from sensors. The inference results and the historical results can be stored in the context knowledge base. The context-aware application can employ the results directly based on different requirements.

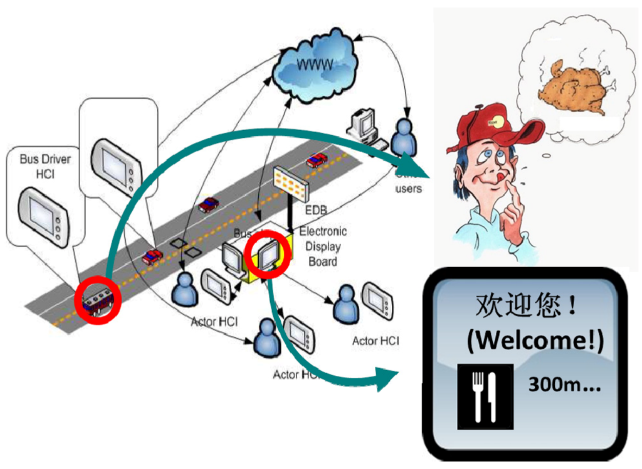
5 Scenario and implementation

In this section, we propose two scenarios (applications) to explain and verify how our intelligent inference engine deals with three facets of problems concerning user activity recognition. The bus stop scenario involves the rules reasoner and the DT reasoner, while the domestic activity application is used to verify the HMM reasoner.

5.1 Bus stop scenario

The bus stop application is a typical application of the Smart City (David et al., 2011). In our previous work, we organised all activities around or in relation to the bus stop shown in Figure 13. The bus stop can provide hot spot services and location-based services. To explain more clearly how context-aware middleware works, we present the bus stop scenario as follows: after an international conference, Tao is taking the bus back to his hotel. He is tired, hungry and only wants to have his favourite meal: roast chicken. However, he has never been to this city before and knows nothing about it. While he is fantasizing about this food, the bus arrives at the bus stop. 'Oh-la-la!' Tao gets off the bus and shouts out with excitement. An avatar, recognising him (by the collection of his identification data) in the large public screen, speaks his native language to him with the subtitles popping up: "Welcome! The roast chicken restaurant is about 300m away from this bus stop. If you want to book a seat, please wave your hand to me…

Figure 13 The bus stop scenario (see online version for colours)



The context-aware middleware collects contextual information from various interaction devices (public screen, etc.), techniques (gesture recognition, markers or face recognition, etc.) and sensors (camera, RFID, QR codes, etc.) for the relevant application. The work process of context-aware middleware for this scenario is as follows: When Tao steps onto a bus and scans his traffic card (RFID card), his public profile is transferred to context-aware middleware at the bus stop via internet. It deduces Tao's current activity: taking the bus. Then, it checks Tao's schedule to find Tao's next activity, having dinner, and searches all the restaurants near the bus stop to recommend him a potential favourite restaurant. The camera fixed at the bus stop distinguishes Tao from the other passengers (face recognition techniques allowed by Tao's profile), and then confirms the reservation by Tao's hand gestures (gesture recognition techniques).

The intelligent inference engine plays an important role in the bus stop scenario. To obtain the user's activity context, it uses two reasoners: the rules reasoner and the DT reasoner. The rules reasoner is responsible for inferring Tao's activity based on a rule set. Therefore, application designers should define first-order logic rules for specific activities, respectively, based on related low-level contexts that can be easily gotten from environments. To simplify understanding, Figure 14 lists some rules that we define for the bus stop scenario.

Figure 14 The sample rules

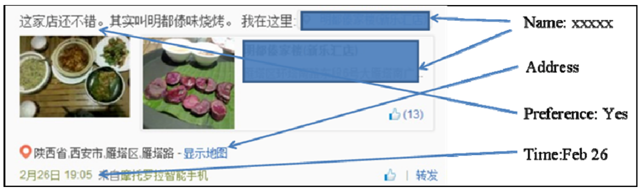
```
@prefix ex: <http://liris.cnrs.fr/smartspace.owl#>.

[Meeting:
        (?user rdf:type ex:User)^
        (?activity rdf:type ex:Meeting)^
        (?activity ex:LocatedIn ex:MeetingRoom2)^
        (?user ex:LocatedIn ex:MeetingRoom2)^
        (?activity ex:startTime ?t1)^
        (?activity ex:endTime ?t2)^
        greaterThan(?currentTime,?t1)^
                        lessThan(?currentimeTime,?t2)

        ->(?user ex:EnagaedIn ?activity)]

[Discussing:
        (?user1 rdf:type ex:Student)^
        (?user2 rdf:type ex:Professor)^
        (?user1 ex:supervisorOf ?user)^
        (?user1 ex:LocatedIn ?Office)^
        (?user1 ex:LocatedIn ?Office)^
        (?user2 ex:hasOffice ?office)

        ->(?user1 ex:Discuss ?user2)]

[TakingBus:
        (?user rdf:type ex:User)^
        (?bus rdf:type ex:Bus)^
        (?user ex:stepOn ?bus)^
        noValue(?user ex:stepOff ?bus)

        ->(?user ex:takeBus ?bus)]

[AtSupper:
        (?user rdf:type ex:User)^
        (?resto rdf:type ex:Restaurant)^
        (?user ex:LocatedIn ?resto)^
        greaterThan(?currentTime,18:30:00)^
        lessThan(?currentimeTime,21:00:00)

        ->(?user ex:Dinning ?resto)]
```

The decision tree focuses on recommending the potential favourite restaurant to Tao. To achieve this recommendation, it requires these three steps: collecting training data, building the decision tree and providing the predictive recommendation.

Collecting training data is one of the most important tasks for the DT reasoner. This requires collecting accurate and available information on the previous activities of each user. This has been considered a tough task in the past, since it is hard to let the user wear diverse sensors to travel around for a long time, only to collect raw training data. The prevalence of social networks provides a possible solution to this problem. Increasingly, people are posting their daily activities on their own social networks as part of life. This applies in particular to the microblog, whose content is typically smaller in both actual and aggregate file size. It is convenient for users to post their activity via mobile devices anywhere and anytime.

We chose the 'weibo' as a data source to collect information on users' activities. (The 'weibo', teeming with more than 300 million users, is the biggest twitter-like microblogging service in China). After analysis, we chose the dinner activity as a research object. We collected available weibo microblogs by a keyword filter, which contains two keyword subsets: one refers to the specific restaurant's name: KFC, McDonalds, etc., whereas the other refers to the set of words usually appearing in the restaurant name: hot pot, restaurant, etc. An example of 'weibo' is shown in Figure 15: it contains the restaurant's name and address, the user's preference and visiting time.

Figure 15 An example of user's Weibo (see online version for colours)



这家店还不错。其实叫明都像味烧烤。我在这里
Name: xxxxx
Address
Preference: Yes
Time:Feb 26
陕西省 西安市 雁塔区 雁塔路 - 显示地图
2月26日 19:05

Detailed quantitative information is necessary for each restaurant, such as price, flavour and environment. The site – www.dianping.com – is opted for as the restaurant's detailed quantitative data source. This is an online independent third-party consumer service rating site, which contains eight sorts of restaurant information: name, price, flavour, service, type, etc., as shown in Figure 16. To process the data in a unified way, the type of restaurant is redefined based on nationality and fast food.

- The user's favourite restaurants from 'weibo', along with their detailed quantitative information construct the training set for choosing a restaurant.
- In this scenario, a real Weibo user's microblog information is used with his permission. We collect the data from 12 May 2011 to 2 May 2012. The clawer has collected 278 available Weibo's microblogs. These chosen restaurants construct a set of training data, some of which are shown in Table 2.
- To build the decision tree, we use J48 implementation of the C4.5 decision tree in Weka (Hall et al., 2009), which is an open source on data mining in Java providing a collection of machine learning algorithms. The learned decision tree providing the profitable suggestion helps the user make a decision relied on the training dataset. As shown in Figure 17, this tree is built to recommend the restaurant for the chosen user.
- The DT reasoner can be invoked based on Tao's schedule. In this scenario, when our context-aware middleware finds that Tao is not in the place where he is scheduled (he is not in a restaurant), it will remind Tao and suggest a choice (a favourite restaurant) based on his current location. The learned decision tree is used to select his favourite restaurant from the list of restaurants located nearby.
- The detailed information for the recommended restaurant is written into an xml file shown in Figure 18(a), and used in the Google map. In this way, the user acquires a restaurant recommendation on the Google map, which helps him/her find this restaurant as shown in Figure 18(b). Various devices, such as a smart phone and pad, can directly access the recommended restaurant information on the map via internet.

Figure 16 Restaurant information from www.dianping.com (see online version for colours)

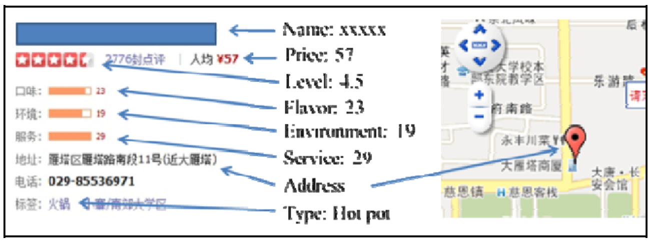


Table 2 Training dataset

No.	*Level*	*Price*	*Flavors*	*Env.*	*Serv.*	*Type*	*Pref.*
1	3.5	39	22	13	13	Chinese	Yes
2	4	15	25	16	17	Chinese	No
3	4	27	19	18	19	Fast food	Yes
4	3	41	18	12	12	Chinese	No
5	5	60	25	25	31	Chinese	Yes
6	3.5	422	16	20	19	Chinese	No
7	5	56	25	23	24	Korea	Yes
8	3.5	55	18	17	15	Japan	No
9	3	24	16	17	15	Fast food	Yes
10	4.5	117	23	26	20	Japan	No
11	4	58	19	20	17	Fast food	Yes
12	4	183	17	24	20	Chinese	No
13	3	33	20	7	11	Chinese	Yes
14	3	7	20	9	10	Chinese	No

5.2 *Domestic activity application*

Furthermore, this subsection will focus on explaining how the HMM reasoner trains and works. We use the domestic activity dataset from van Kasteren et al. (2008) to verify the HMM reasoner. They employed 14 binary input sensors to record a user's seven kinds of daily activities: leaving the house, using the toilet, taking a shower, going to bed, preparing breakfast, preparing dinner and drinking in an apartment as shown in Figure 19 from 25 February 2008 to 21 March 2008.

This annotated real-world dataset contains 245 activities and the corresponding binary input state as shown in Figure 20.

Figure 17 The learned decision tree for choosing a restaurant (see online version for colours)

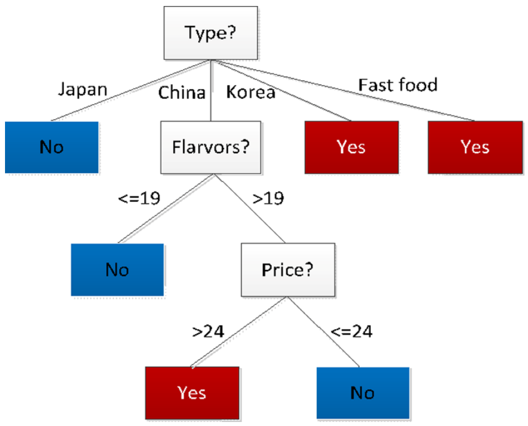
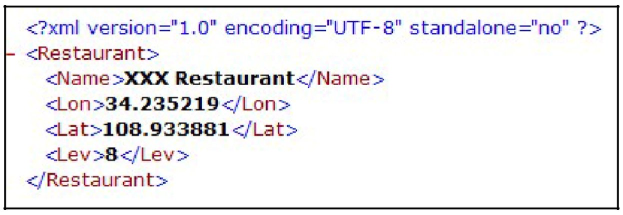
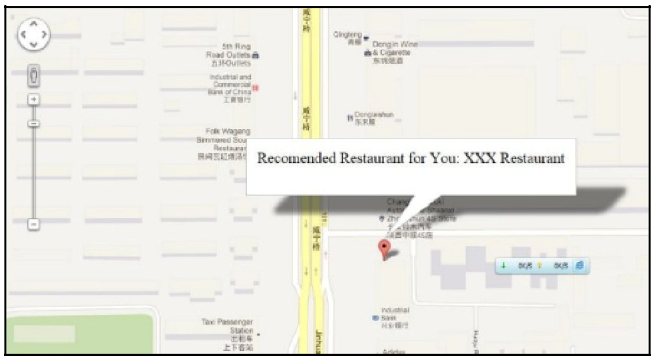


Figure 18 The DT reasoner results: (a) detailed information in xml and (b) recommended results in Google map (see online version for colours)



```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <Restaurant>
    <Name>XXX Restaurant</Name>
    <Lon>34.235219</Lon>
    <Lat>108.933881</Lat>
    <Lev>8</Lev>
  </Restaurant>
```

(a)



(b)

This task is split into two parts: estimating process parameters based on previous data (training data) and using these parameters to infer the real-world process by looking at the novel sensor reading. On the basis of Lim and Dey's (2010) work, we trained an HMM with a sequence length of 5 min, and 1 min per sequence step. The Baum-Welch algorithm (Baum et al., 1970) is used for training. In this paper, we will not describe the train process. Detailed information on the HMM can be referred to in Rabiner (1989).

Figure 19 The floor plan of the test apartment (see online version for colours)

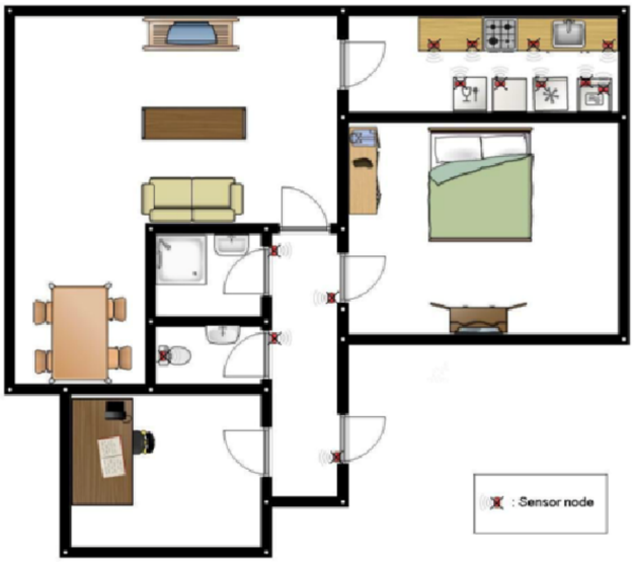
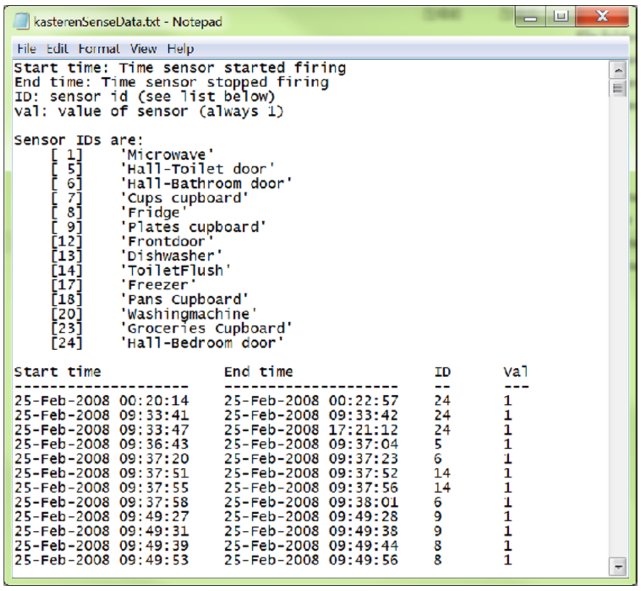


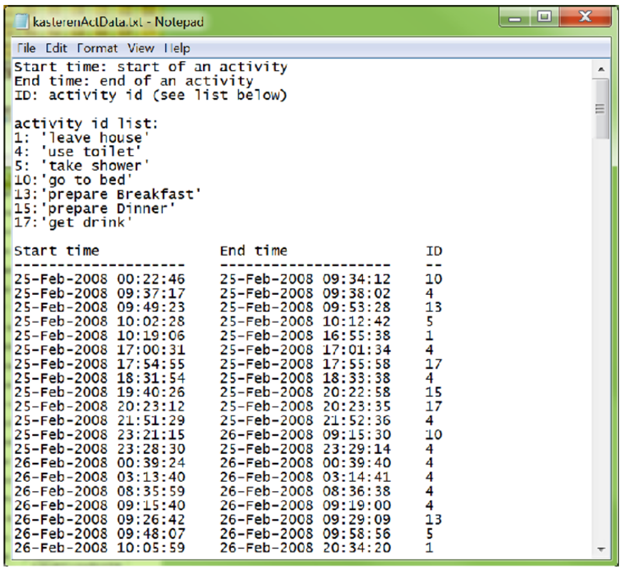
Figure 20 The domestic activity dataset: (a) the state of 14 binary input sensors and (b) user's activities (see online version for colours)



```
Start time: Time sensor started firing
End time: Time sensor stopped firing
ID: sensor id (see list below)
val: value of sensor (always 1)

Sensor IDs are:
     [ 1]     'Microwave'
     [ 5]     'Hall-Toilet door'
     [ 6]     'Hall-Bathroom door'
     [ 7]     'Cups cupboard'
     [ 8]     'Fridge'
     [ 9]     'Plates cupboard'
     [12]     'Frontdoor'
     [13]     'Dishwasher'
     [14]     'ToiletFlush'
     [17]     'Freezer'
     [18]     'Pans Cupboard'
     [20]     'Washingmachine'
     [23]     'Groceries Cupboard'
     [24]     'Hall-Bedroom door'

Start time              End time                ID      Val
--------------------    --------------------    --      ---
25-Feb-2008 00:20:14    25-Feb-2008 00:22:57    24      1
25-Feb-2008 09:33:41    25-Feb-2008 09:33:42    24      1
25-Feb-2008 09:33:47    25-Feb-2008 17:21:12    24      1
25-Feb-2008 09:36:43    25-Feb-2008 09:37:04    5       1
25-Feb-2008 09:37:20    25-Feb-2008 09:37:23    6       1
25-Feb-2008 09:37:51    25-Feb-2008 09:37:52    14      1
25-Feb-2008 09:37:55    25-Feb-2008 09:37:56    14      1
25-Feb-2008 09:37:58    25-Feb-2008 09:38:01    6       1
25-Feb-2008 09:49:27    25-Feb-2008 09:49:28    9       1
25-Feb-2008 09:49:31    25-Feb-2008 09:49:38    9       1
25-Feb-2008 09:49:39    25-Feb-2008 09:49:44    8       1
25-Feb-2008 09:49:53    25-Feb-2008 09:49:56    8       1
```

(a)

Figure 20 The domestic activity dataset: (a) the state of 14 binary input sensors and (b) user's activities (see online version for colours) (continued)



```
kasterenActData.txt - Notepad
File Edit Format View Help
Start time: start of an activity
End time: end of an activity
ID: activity id (see list below)

activity id list:
1: 'leave house'
4: 'use toilet'
5: 'take shower'
10:'go to bed'
13:'prepare Breakfast'
15:'prepare Dinner'
17:'get drink'

Start time               End time               ID
--------------------     --------------------   --
25-Feb-2008 00:22:46     25-Feb-2008 09:34:12   10
25-Feb-2008 09:37:17     25-Feb-2008 09:38:02   4
25-Feb-2008 09:49:23     25-Feb-2008 09:53:28   13
25-Feb-2008 10:02:28     25-Feb-2008 10:12:42   5
25-Feb-2008 10:19:06     25-Feb-2008 16:55:38   1
25-Feb-2008 17:00:31     25-Feb-2008 17:01:34   4
25-Feb-2008 17:54:55     25-Feb-2008 17:55:58   17
25-Feb-2008 18:31:54     25-Feb-2008 18:33:38   4
25-Feb-2008 19:40:26     25-Feb-2008 20:22:58   15
25-Feb-2008 20:23:12     25-Feb-2008 20:23:35   17
25-Feb-2008 21:51:29     25-Feb-2008 21:52:36   4
25-Feb-2008 23:21:15     26-Feb-2008 09:15:30   10
25-Feb-2008 23:28:30     25-Feb-2008 23:29:14   4
26-Feb-2008 00:39:24     26-Feb-2008 00:39:40   4
26-Feb-2008 03:13:40     26-Feb-2008 03:14:41   4
26-Feb-2008 08:35:59     26-Feb-2008 08:36:38   4
26-Feb-2008 09:15:40     26-Feb-2008 09:19:00   4
26-Feb-2008 09:26:42     26-Feb-2008 09:29:09   13
26-Feb-2008 09:48:07     26-Feb-2008 09:58:56   5
26-Feb-2008 10:05:59     26-Feb-2008 20:34:20   1
```

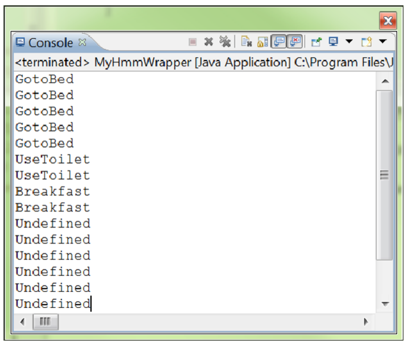
(b)

The application takes 14 binary input sensors and infers which activity (out of seven) the user is performing. As we focus only on verifying the HMM reasoner, sensor information is simulated in this example. We simulate a test sensor data sequence, which is represented by a 15×14 matrix shown in Figure 21(a). The row represents the situation of 14 sensors, whereas the column represents the time slice. The data sequence is placed in context-aware middleware successively. Then, the learned HMM (parameters determined) reasoner carries out inference of the activity sequence by calculating its probability given an observation sequence (sensor data sequence). The Viterbi (1967) algorithm is used to infer. The entire HMM process is implemented based on jhmm in Java, and the final result is shown in Figure 21(b). As mentioned earlier, the HMM ruler should work all the time. These results and the historical results are stored in the context knowledge base, which can be used by various context-aware applications via our context-aware middleware.

Figure 21 The HMM results for domestic activity: (a) the simulated input data and (b) the printed results (see online version for colours)

$$\begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&1&0&0&0&0&1\\
0&0&0&0&0&1&0&0&0&0&0&0&1&1\\
0&0&0&0&1&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1
\end{bmatrix}$$

(a)



```
Console
<terminated> MyHmmWrapper [Java Application] C:\Program Files\J
GotoBed
GotoBed
GotoBed
GotoBed
GotoBed
UseToilet
UseToilet
Breakfast
Breakfast
Undefined
Undefined
Undefined
Undefined
Undefined
Undefined
```

(b)

6 Related work

The context-aware system is an answer to challenges associated with service discovery, mobility, environmental changes and context retrieval (Romero et al., 2008). The Active Badge System (Want et al., 1992) is commonly considered as the first research investigation into context awareness. In this work, context information refers primarily to location. From then on, a large number of infrastructures have provided services for handling context. Context inference always plays an important role in these systems. The context toolkit, developed by Dey et al. (2001), is a toolkit that supports development of context-aware applications. The context interpreter part is used to deduce high-level context information from low-level information. CASS (Fahy and Clarke, 2004) supports context-aware applications on hand-held computing devices and other small mobile computing devices. The important feature of CASS is that it supports abstraction of high-level context and separate context based on inferences and behaviours from the application code. CoBrA (Chen et al., 2003) is one of the earliest systems using semantic web technology to support context-aware pervasive computing. The CoBrA inference engine is used in both types of reasoning. Besides detecting and resolving inconsistent knowledge, it can also infer context knowledge that cannot be easily acquired from the physical sensors. The context reasoner in SOCAM (Gu et al., 2004) supports two kinds of reasoning: ontology reasoning and user-defined rule-based reasoning. Truong and Dustdar (2009) have summarised inference techniques supported by existing systems. However, they found that context inference is not well suited and that most systems are simply based on semantic reasoning.

7 Conclusion

In recent work, more and more research revolves around the 'activity' context as opposed to the 'location' context. We reviewed literature on activity context recognition in three premier conferences held on context awareness in the last 10 years, summarised all the methods and divided research concerning activity context recognition into three main facets: basic activity inference, dynamic activity analysis and future activity recommendation. On the basis of our previous work, we proposed an intelligent inference engine for context-aware middleware, consisting of a basic inference module and an intelligent inference module. Besides satisfying requirements for checking context consistency, our inference engine integrates the three most popular methods for activity context recognition: rules, decision tree and HMM. In addition, we designed a mechanism for organising three algorithms to support expandability and scalability for the intelligent inference engine, and to provide different invoking mechanisms according to different reasoner tasks. This provides a solution for all facets of activity context recognition based on our context-ware middleware. Finally, the two scenarios are presented to describe how the inference engine works.

With respect to the bus stop scenario, we extract a user's activity context from his social networks as a training set with the user's permission, to investigate the rules reasoner and the DT reasoner. Concerning domestic activity, we adopt the domestic activity dataset to verify the HMM reasoner. However, intelligent inference failed to take into consideration the context ambiguity problem, i.e., contexts from the sensors are not always correct. The context inference engine will reach the conclusion based on inaccurate information, thus giving rise to incorrect actions by the application, especially in the rules reasoner. We will improve the context inference in this field to make our system more robust and intelligent.

References

Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M. and Steggles, P. (1999) 'Towards a better understanding of context and context-awareness', *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, HUC '99*, Springer-Verlag, London, UK, pp.304–307.

Baldauf, M., Dustdar, S. and Rosenberg, F. (2007) 'A survey on context-aware systems', *Int. J. Ad Hoc Ubiquitous Comput.*, Vol. 2, pp.263–277.

Ballendat, T., Marquardt, N. and Greenberg, S. (2010) 'Proxemic interaction: designing for a proximity and orientation-aware environment', *ACM International Conference on Interactive Tabletops and Surfaces, ITS '10*, ACM, New York, NY, USA, pp.121–130.

Baum, L., Petrie, T., Soules, G. and Weiss, N. (1970) 'A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains', *Ann. Math. Stat.*, Vol. 41, pp.164–171.

Chen, H., Finin, T. and Joshi, A. (2003) 'An intelligent broker for context-aware systems', *Adjun. Proc. Ubicomp*, pp.12–15.

David, B., Zhou, Y., Xu, T. and Chalon, R. (2011) 'Mobile user interfaces and their use in a smart city', *WorldComp'11*, Las Vegas, Nevada, USA, pp.383–388.

Dey, A.K., Abowd, G.D. and Salber, D. (2001) 'A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications', *Hum-Comput. Interact*, Vol. 16, pp.97–166.

Fahy, P. and Clarke, S. (2004) 'CASS – a middleware for mobile context-aware applications', *Workshop on Context Awareness, MobiSys*, Boston, USA, pp.304–308.

Geib, C.W., Maraist, J. and Goldman, R.P. (2008) 'A new probabilistic plan recognition algorithm based on string rewriting', *ICAPS*, pp.91–98.

Gu, T., Pung, H.K. and Zhang, D.Q. (2004) 'A middleware for building context-aware mobile services', *Vehicular Technology Conference, 2004, VTC 2004-Spring, 2004 IEEE 59th.*, pp.2656–2660.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Witten, I.H. (2009) 'The WEKA data mining software: an update', *SIGKDD Explor Newsl.*, Vol. 11, pp.10–18.

Hu, D.H., Zheng, V.W. and Yang, Q. (2011) 'Cross-domain activity recognition via transfer learning', *Pervasive Mob. Comput.*, Vol. 7, pp.344–358.

Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H. and Malm, E-J. (2003) 'Managing context information in mobile devices', *IEEE Pervasive Comput.*, Vol. 2, pp.42–51.

Krumm, J. (Ed.) (2009) *Ubiquitous Computing Fundamentals*, 1st ed., Chapman and Hall/CRC, London.

Lim, B.Y. and Dey, A.K. (2010) 'Toolkit to support intelligibility in context-aware applications', *Proceedings of the 12th ACM International Conference on Ubiquitous Computing, Ubicomp '10*, ACM, New York, NY, USA, pp.13–22.

Lin, L. (2006) *Location-Based Activity Recognition*, PhD Thesis, University of Washington.

Lukowicz, P., Pentland, S. and Ferscha, A. (2012) 'From context awareness to socially aware computing', *IEEE Pervasive Comput.*, Vol. 11, pp.32–41.

Pazzani, M., Muramatsu, J. and Billsus, D. (1996) 'Syskill & Webert: Identifying interesting web sites', *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Vol. 1, AAAI'96, AAAI Press, pp.54–61.

Pollack, M.E., Brown, L., Colbry, D., McCarthy, C.E., Orosz, C., Peintner, B., Ramakrishnan, S. and Tsamardinos, I., (2003) 'Autominder: an intelligent cognitive orthotic system for people with memory impairment', *Robotics and Autonomous Systems*, Vol. 44, pp.273–282.

Rabiner, L. (1989) 'A tutorial on hidden Markov models and selected applications in speech recognition', *Proc. IEEE 77*, pp.257–286.

Romero, D., Parra, C., Seinturier, L., Duchien, L. and Casallas, R. (2008) 'An SCA-based middleware platform for mobile devices', *Enterprise Distributed Object Computing Conference Workshops, 2008 12th. Presented at the Enterprise Distributed Object Computing Conference Workshops*, 12th, pp.393–396.

Schmidt, A., Beigl, M. and Gellersen, H. (1998) ‚There is more to context than location', *Comput. Graph.*, Vol. 23, pp.893–901.

Stojanovic, D. (2009) 'Context-aware mobile and ubiquitous computing for enhanced usability: adaptive technologies and applications (premier reference source)', *Information Science Reference*, 1st ed., Hershey, PA.

Truong, H-L. and Dustdar, S. (2009) 'A survey on context-aware web service systems', *Int. J. Web Inf. Syst.*, Vol. 5, pp.5–31.

Vail, D.L., Veloso, M.M. and Lafferty, J.D. (2007) 'Conditional random fields for activity recognition', *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '07*, ACM, New York, NY, USA, pp.235:1–235:8.

Van Kasteren, T., Noulas, A., Englebienne, G. and Kröse, B. (2008) 'Accurate activity recognition in a home setting', *Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp '08*, ACM, New York, NY, USA, pp.1–9.

Viterbi, A.J. (1967) 'Error bounds for convolutional codes and an asymptotically optimum decoding algorithm', *IEEE Trans. Inf. Theory*, Vol. 13, pp.260–269.

Wang, X.H., Zhang, D.Q., Gu, T. and Pung, H.K. (2004) 'Ontology based context modeling and reasoning using OWL', *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, PERCOMW '04*, IEEE Computer Society, Washington, DC, USA, pp.18–23.

Want, R., Hopper, A., Falcão, V. and Gibbons, J. (1992) 'The active badge location system', *ACM Trans. Inf. Syst*., Vol. 10, pp.91–102.

Xu, T., David, B., Chalon, R. and Zhou, Y. (2011) 'A context-aware middleware for ambient intelligence', *Proceedings of the Workshop on Posters and Demos Track, PDT '11*, ACM, New York, NY, USA, pp.10:1–10:2.