Supporting Activity Context Recognition in Context-Aware Middleware

Tao Xu, Yun Zhou, Bertrand David, René Chalon

Université de Lyon, CNRS,

Ecole Centrale de Lyon, LIRIS, UMR5205

{tao.xu, yun.zhou, bertrand.david, rene.chalon }@ec-lyon.fr

Abstract

Context-aware middleware is considered as an efficient solution to develop context-aware application, which provides a feasible development platform integrating various sensors and new technologies. With the development of sensors, the research work on the context shift from "location" to "activity" gradually. Then it puts forward a new requirement for context-aware middleware: activity context recognition. In this paper, we first survey the related literature in premier conferences over the past decade, review the main methods of activity context recognition, and conclude its three main facets: basic activity inference, dynamic activity analysis and future activity recommendation. We then propose an intelligent inference engine integrating three popular methods solving three facets mentioned above respectively, which is based on our context-aware middleware. Finally, to evaluate the feasibility of our proposal, we implement two scenarios: bus stop scenario and domestic activity application.

Introduction

The popularity of mobile smart devices, the emergence of various powerful sensors, the booming of wearable computing devices, they all declare an advent of new era: Ubiquitous computing. Context-aware computing, as the central theme in Ubiquitous computing (Krumm 2009), draws more and more researchers' attentions. Context-aware systems are concerned with the acquisition of context (e.g. using sensors to perceive a situation), the abstraction and understanding of context (e.g. matching a perceived sensory stimulus to a context), and application behaviors based on the recognized context (e.g. triggering actions based on context)(Schmidt, Beigl, and Gellersen 1998). To reach this aim, it needs to integrate all sensors, actuators, communication objects and computing devices to the system. Low level mechanisms and drivers are necessary but they must be encapsulated to more common and higher level view allowing to collect, treat and propagate the information between these components in an appropriate manner, i.e. take into account the changes in the context, and propagate appropriate decisions (Xu et al. 2011). Context-aware middleware provides a feasible solution.

Context inference plays a role of the brain in the context-aware middleware, which contains two main tasks: one is checking context's consistency; another one is determining or inferring the user's situation. Once the user's situation has inferred, the application can take an appropriate action (Krumm 2009). In recent year, most of context-aware middleware have adopted the ontology-based model. In this model, OWL is responsible for checking context's consistency. The research focuses are shifted to infer the user's situation (commonly refer to activity). The existing context-aware middleware usually employ rule-based method to solve this problem. The rule-based method is relatively easy to build and relatively intuitive to work with. However, it is fragile, and not enough flexible. It is inadequate to support diverse types of tasks. The holy grail of the context awareness is to divine or understand human intent (Krumm 2009). It requires a much smarter "brain" to deal with various issues. Based on this point, context inference is expected to support versatile activity context recognition in context-aware computing.

In this paper, we review all the methods on activity context recognition published in three premier conferences in past decade and conclude that activity context recognition is divided into three facets: basic activity inference, dynamic activity analysis and future activity recommendation. Then we propose an intelligent inference engine based on our context-aware middleware. It integrates three most popular methods of activity context recognition used in context-aware application, and provides a solution to meet different requirements in activity context recognition.

In the remainder of this paper, a widely acceptable definition of context is presented, along with discussion of main types of context. Following these discussions, we review work in activity context recognition published in premier conferences in the past decade. Then, we present our context-aware middleware, along with detailed intelligent inference engine. To go one step further, two scenarios are introduced to explain how it works. Finally, we conclude our work, provide the pros and cons of our intelligent inference engine and discuss our further direction.

Context and Activity Context Recognition

Context is one of the critical concepts in context-aware computing. The widely used definition of context is proposed by Day and Abowd, and the statement is as follows (Abowd et al. 1999):

"Context is any information that can be used to characterize 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."

With respect to context-aware computing, they focus on *who's, where's, when's* and *what's* (that is, what activities are occurring) of entities and use this information to determine why a situation is occurring (Krumm 2009). The research work on the context awareness is simplified to achieve the five "*w*". Among these "*w*", the "*why*" is the destination and other four "*w*" (*who, where, when, what*) represent four kinds of fundamental context: user, location, time, and activity. With the rapid development of technology and widespread use of smart mobile devices and sensors, the first three fundamental contexts (user, location and time) become to be relative easily captured directly from the context sources nowadays. However, recognizing user's activities has been still a tough task.

Activity context recognition aims to infer a user's behavior from observations such as sensor data (Hu, Zheng, and Yang 2011), and has various applications including medical care (Pollack et al. 2003), logistics service (Lin 2006), robot soccer (Vail, Veloso, and Lafferty 2007), plan recognition (Geib, Maraist, and Goldman 2008), etc. Many researchers make great attempts to find efficient way to recognize user's activities. Continuing the Lim and Dey's work (Lim and Dey 2010), we reviewed literature from three top-tier conferences on context-aware computing in recently years, which contain 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, the International Conference on Pervasive Computing **(Pervasive)** from 2004 to 2012. There are 59 papers related to user's activity recognition, which involve with seven algorithms: Rules, Decision Tree (DT), Naïve Bayes (NB), Hide Markov Model (HMM), Support Vector Mechanic (SVM), k-Nearest Neighbor Algorithm (kNN), and Artificial Neural Networks (ANNs). The detailed statistical results are shown in Figure 1.

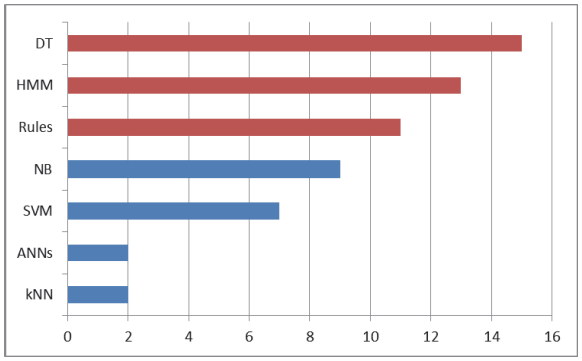


Figure.1.The Statistical Results of Activity Recognition's Algorithm

We conclude researches on recognizing user's activity into three facets shown in Figure 2: basic activity inference, future activity recommendation, and dynamic activity analysis.

- **Basic activity inference** focuses on recognizing 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 basic activity.
- **Future activity recommendation** usually refers to recommend or predict user's future activity or choices. To achieve it, besides taking into consideration user's current situation (context), it requires analyzing the context of user's pervious similar behaviors (training data). Decision tree is adequate to tackle with this problem. It can be considered as a type of context-aware recommendation.
- **Dynamic activity analysis** is responsible for analyzing fine-grained activity compared with basic activity inference. There are two main solutions: one employs State-Space Model, which partitions the activity into state sequences and infers the type of activity (state sequence) based on probability of an observed sequence such as Bayesian Networks and Hidden Markov Model. The other one relies on pattern recognition techniques, which extracts patterns from activities and infers the type of activity based on different

activity pattern, like Vector Support Machine, Artificial Neural Network, etc. Being similar with future activity recommendation, it requires learning the process parameters from the previous activity information as well.

We can observe that the top tree popular methods are Rules, DT, HMM. They have their own advantages and characteristics. We analyze them (the detailed information will be stated in followed section) and found that the three methods can be leveraged to deal with these three facets respectively. As well these three methods mostly represent the mainstream solution on their actual facet. We integrate the three methods: Rule, DT, and HMM into our context-aware middleware as our research objective.

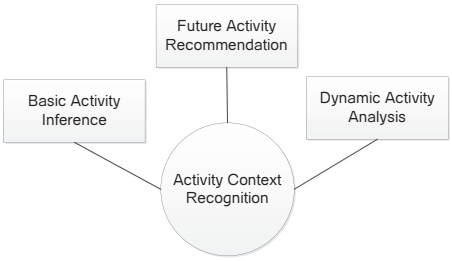


Figure.2.The Three Main Facets of Activity Context Recognition

Context-aware Middleware

Context-aware applications are becoming more prevalent. It can be found in the areas of wearable computing, mobile computing, robotics, adaptive and intelligent user interfaces, augmented reality, adaptive computing, intelligent environments, and context-sensitive interfaces (Krumm 2009). However, development of context-aware applications is still inherently complex. These applications require adapting to change context information: physical context, computational context, and user context/tasks (Bettini et al. 2010). To rapid prototyping of context-aware services in Ambient Intelligence, we have designed a context-aware middleware (Xu et al. 2011) (Xu et al. 2013) based on MOCOCO (Mobility, Contextualization and Cooperation) (B. T. David and Chalon 2007).

The context-aware middleware is shown in Figure 3, which is organized in two layers: the low layer is an Enterprise Service Bus, which provides a solution to integrate sensors and actuators with a standardized data representation and unified standard interface to achieve the core functions of service interaction: service registry, service discovery and service consumption.

The versatile context interpreter is the high layer, which is in charge of context inferences, expressive query, and persistent storage. It is consisted of four parts:

The context aggregator is responsible for working with basic contextual data collected by ESB to carry out fusion and fission of information.

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 has two main tasks: The one is to handle queries from the application. It supports SPARQL, which is an RDF query language, able to retrieve and manipulate data stored in OWL. The other one is to invoke the context inference engine. When the application needs 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 context-aware middleware. Its detailed information will be presented in next section.

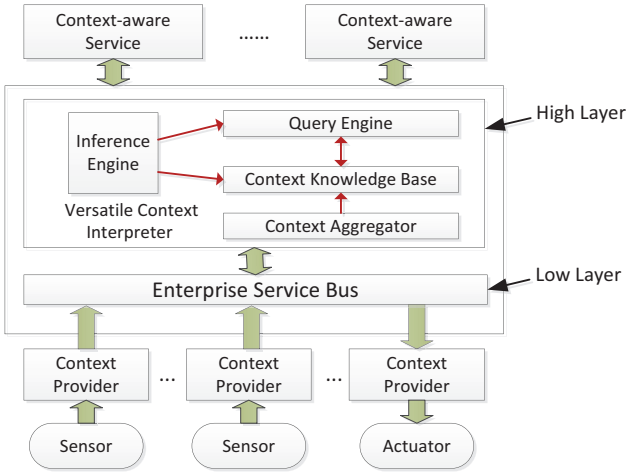


Figure.3.The Architecture of Context-aware Middleware

The Intelligent Inference Engine

Towards a better understanding of human intent, we propose an intelligent inference engine, which is composed 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.

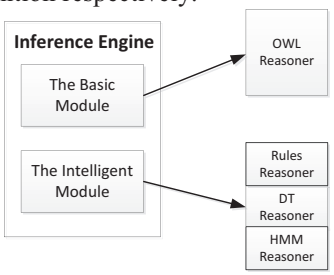


Figure.4.The Structure of Inference Engine

The basic module's main task is checking context consistency. We adopt the ontology-based context model to construct our context-aware system. It has good features for developing context-aware system, such as knowledge sharing, knowledge reuse, logic inference. Also it lays the ground work for basic inference module.

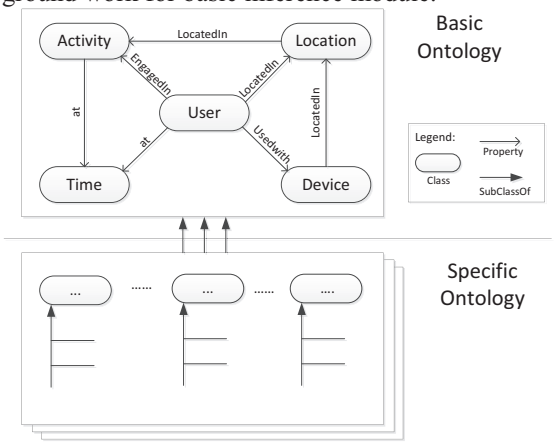


Figure.5.Context expression ontology

For the context model, we employ a hierarchical structure to describe the user's situation and circumstance based on Web Ontology Language (OWL), which is an ontology markup language adopted by W3C as the standard of semantic web. The structure is shown in Figure 5. The basic model defines generic conceptions and relationships in context-aware computing, which comes up with a basic context structure. It has five interrelated basic classes: user, location, time, activity, and device, representing who, where, when, what activities, and devices, and seven properties (relationships) between classes. General 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. The general context-aware model is developed by context-aware middleware designers, and then the specific model is developed by context-aware application developers.

OWL has a built-in reasoner based on the description logic. This reasoner can fulfill the essence of logical requirements, which comprises concept satisfaibility, class subsumption, class consistency, and instance checking. The table below shows a partial basic reasoning rules.

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 supporting activity context recognition. According to aforementioned reviews on methods of activity recognition, we integrate the most popular methods: Rules, DT, HMM to provide a solution to three facets of activity context recognition: basic activity inference, dynamic activity analysis and future activity recommendation.

Rules

"Rules" is the early attempt to infer user's activity in context-aware computing. "Rules" refers to the method using a set of *if-then* rule to infer user's activity based on the first order logic: *if* the devices sense a particular situation, *then* it can deduce user's activity. "Rules" is made based on general features of activity. These chosen features of activity, namely low-level context, should be able to be obtained by physical sensors or other context sources. We employ a simple example as shown below to interpret it.

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

Figure.6.This is an Example of Rules

"Rules" has a wide range of adaptabilities, which is usually designed to provide inference for almost all users. Furthermore it is relative intuitive and easy to work with. However, rules are rigid, meaning that they are also brittle (Krumm 2009). Even trifling exceptions will cause some errors. Since this method is easy to be implemented in context-aware application, it is an effective and widespread used method to infer user's activity. We leverage it to work on basic activity inference.

Besides Rules, the common alternative approaches involve with artificial intelligence algorithms. DT and HMM both belong to this category. They have their own characters to focus on different facets.

Decision Tree (DT)

Decision Tree is a classic algorithm in the field of artificial intelligence. It is a predictive model which maps observations on an item to conclude about 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. The processing for partitioning data is running on the values of

items' attributes. The choice of the item's 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 measure of the expected encoding length measured in bit.

$$Entropy(\mathrm{S}) = \sum -p(\mathrm{s}) \log_2 p(\mathrm{s}) \quad (1)$$

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:

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

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 value v. Information gain is used to select the best attribute at each step in growing the tree. More clearly, the classic example is shown below.

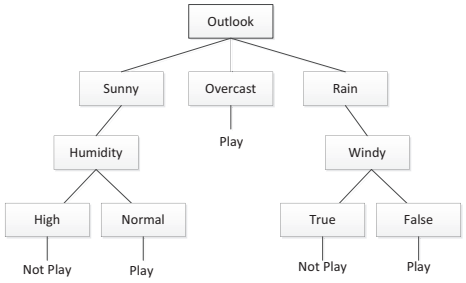


Figure.7.The Classic Example of Decision Tree

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

Hide Markov Model (HMM)

The HMM is the most accepted algorithm in temporal recognition tasks including speech, gesture, activity, etc. HMM is a statistical Markov model which can recover a data sequence that is not immediately observable. In human activity recognition, complex activities have temporal structure. The time series data obtained by sensors is divided into time slices of constant length and each slice is labeled with state of activity. A generic HMM for activity is illustrated as shown in Figure 8. The shaded nodes represent observable variables (data from sensors), while the white nodes represent hidden ones (state of activity). P is the state transition probabilities, Q is observation probability.

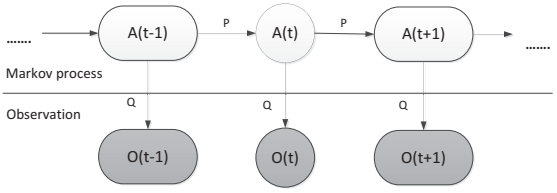


Figure.8.The Simple Example of HMM for Activity Recognition

HMM can learn the parameters "P, Q" of probabilistic model from the training data. Inference which best labeled sequence explaining the new coming data from the sensorsis depended on calculating a maximum of the conditional probability $P(a_1, a_2, a_3 ... | \mathrm{o}_1, \mathrm{o}_2, \mathrm{o}_3 ...)$:

$$a_1, a_2, a_3 ... = \underset{all\ a_1, a_2, a_3 ...}{ArgMax}\, P(a_1, a_2, a_3 ... | \mathrm{o}_1, \mathrm{o}_2, \mathrm{o}_3 ...) \quad (3)$$

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.

Organization of Three Algorithms

We adopt the strategy pattern to organize the three algorithms. It encapsulates the algorithm into separated classes, which enable the context-aware application developer to vary the algorithm independently of the context and to plug in a new one at runtime. The strategy pattern offers an alternative to conditional statements for selecting desired behavior, which makes the three algorithms interchangeable. It gives this module flexibility, so that the context-aware application developers can alter and extend the module, such as adding other algorithm without affecting the originals, as long as the algorithms employing the correct "InferBehavior" interface.

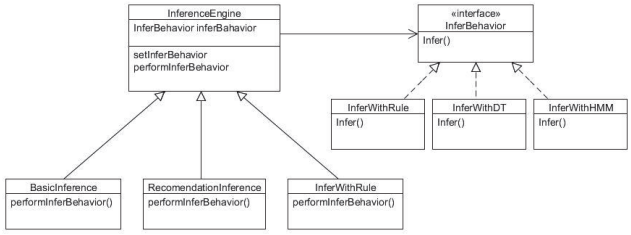


Figure.9.The Partial UML of the Inference Engine

We provide a universal interface. If the context-aware application developers want to adopt different inference methods to deal with different situation, they just invoke the function with same name based on this interface, do not

need to involve into the actual algorithm. The partial UML model is shown in Figure 9.

The Invoking Mechanism

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

In the basic module, OWL reasoner's main task is to check context consistent, it is invoked when the new context model is added in, such as adding a new specific model. In the intelligent module, due 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 enable rules reasoner to work when receiving the query from external service, such as context-aware applications. The Rules reasoner works as the basic inference using most frequently. The context-aware application developers formulate the different rule sets for different activities. Based on efficiency consideration, the rules reasoner will work on the different rule sets according to queries relating to types of activities.

With respect to the DT reasoner, we present a schedule invoking mechanism, which invokes the DT reasoner based on user's schedule. The DT reasoner is in charge of future activity recognition, which works with user's schedule. When the system finds that the user doesn't in the place where user is scheduled, the system reminds user and gives a suggested choice based on user's current location. In the bus stop scenario, we are focusing on providing the appropriate near-located restaurant to user.

For the HMM reasoner, we enable it to work always when the middleware system is performing, due to the requirement of reasoner's algorithm and task. It provides the activity sequence inference based on the sequence sensors data. Thus it requires time series data from sensors. The inference results and the historical results can be stored in context knowledge base. The context-aware application can employ the results directly based on different requirements.

Scenario and Implementation

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

Bus Stop Scenario

Bus stop application is a typical application of Smart City (B. David et al. 2011). In our previous work, we organize all activities around or in relation with the bus stop. The bus stop can provide hot spot services and located-based services. To explain better how the context-aware middleware works, we present bus stop scenario as follows: After a conference, Tao is taking the bus back to his hotel. He is tired, hungry and only wishes to have his most favorite meal: the roast chicken. However, he has never been to this city before and knows nothing about it. When he is lost in fantasy of the food, the bus arrives at the bus stop. "Oh-la-la!" Tao takes off the bus and shouts out with excitement. An avatar, recognizing 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 far from this bus stop. If you want to book a place, please wave your hand to me…

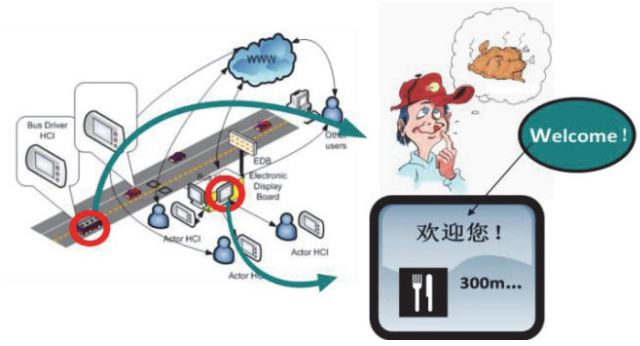


Figure.10.The Bus Stop Scenario

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.) to relevant application. The work process of context-aware middleware for this scenario is as follows: When Tao steps on to a bus and scan his traffic card (RFID card), his public profile is transferred to context-aware middleware in bus stop via internet. It deduces the Tao's current activity: taking 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 a potential favorite restaurant to him. The camera fixed in bus stop distinguishes Tao from the other passengers (face recognition techniques allowed by Tao's profile), and then confirm the reservation by Tao's hand gestures (gesture recognition techniques).

The intelligent inference engine plays an important role on this case. Rules reasoner is responsible for inferring the Tao's activity based on rule set. In this scenario, the Tao's ongoing activity (taking bus) is deduced on "*if* user steps

on a bus, and does not takes it off, *then* user is taking bus". Rules reasoner is invoked when applications query tao's situation. It is implemented by Jena, a Java framework for building Semantic Web application, which comes up with inference on the first order logic. Developers can freely define the rules for the different specific situation.

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

Collecting the training data is one of the foremost tasksfor DT reasoner. It requires collecting accurate and available information of pervious activities of every user. It has been considered as a tough task in the past, since it hard to let the user wears diverse sensors to travel around during a long time only for collecting the raw training data. The prevalence of social networks provides a possible solution for it. Increasing people get used to post their daily activities on their own social networks as parts of life. Especially the microblog, its content is typically smaller in both actual and aggregate file size; it is a convenient way for user to post their activity by mobile device everywhere.

We choose the "weibo", as data source to collect information of users' activities. (The "weibo", teeming with more than 300 million users, is the biggest twitter-like microblogging service in China). After the analysis, we chose dinner activity as a research object. We collect available weibo's micro blogs by a keywords filter, which contains two keywords subsets: the one refers to the specific restaurant's name: KFC, Mcdonalds, etc. and the other one refers to the set of word usually appearing in restaurant name: hot pot, restaurant, etc.

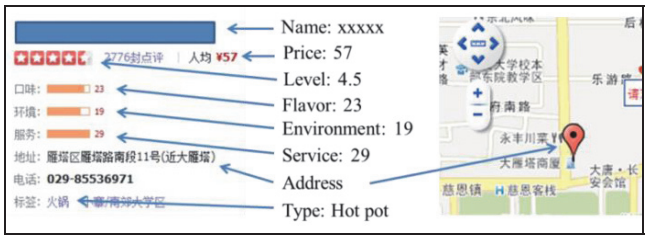


Figure.11.The Information of Restaurant from www.dianping.com

Each restaurant's detailed quantitative information is necessary, such as price, flavor, and environment. The site: www.dianping.com is opted for the restaurant's detailed quantitative data source. It is an online independent third-party consumer service rating sites, which contains eight sorts of restaurant information: name, price, flavor, service, type, etc., as shown in Figure 11. For convenience, type of restaurant is redefined based on nationality and fast food.

User's favorite restaurants from "weibo", along with their detailed quantitative information construct the training set for choosing restaurant.

In this scenario, a real weibo user's micro blog information is adopted under his permission. We collect the data from May 12th, 2011 to May 2nd, 2012. The clawer has collected 14 available weibo's micro blog about restaurant. These chosen restaurants construct a set of training data, as shown in the table 2.

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

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 to provide a collection of machine learning algorithms. The learned decision tree to provide the profitable suggestion to help the user to make a decision relied on the training dataset. As shown in Figure 12, this tree is built to recommend restaurant for the chosen user.

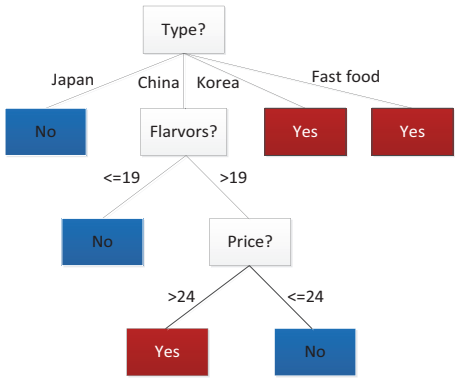


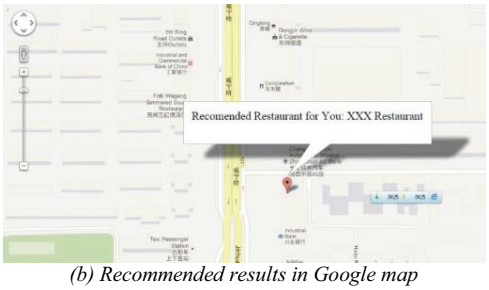
Figure.12.The Learned Decision Tree for Choosing Restaurant

DT reasoner is 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 give a suggested choice (a favorite restaurant) based on his current location. The learned decision tree is used to select the favorite restaurant from the near located restaurants list.

The detailed information of the recommended restaurant is written into an xml file and is used in Google map. In this way, the user gets a restaurant recommendation on the Google map, which helps to find this restaurant, as shown in Figure 13. Various devices, like smart phone, pad etc. can directly access the recommended restaurant on map via internet.

```
<?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) Detailed information in xml



(b) Recommended results in Google map

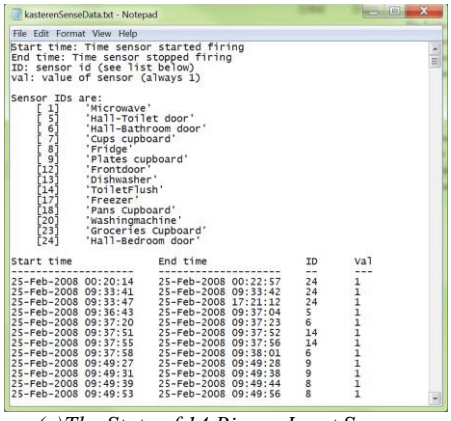
Figure.13.The Results for DT Reasoner

Domestic Activity Application

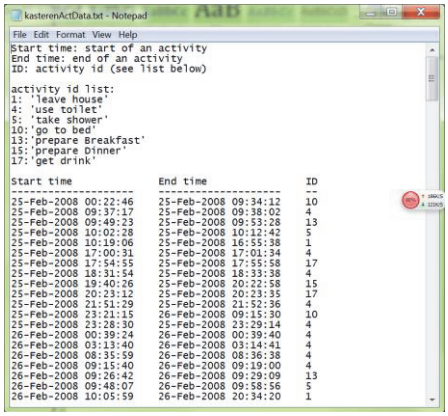
We use the domestic activity dataset from (van Kasteren et al. 2008) to verify HMM reasoner. They employed 14 binary input sensors to record a user's seven kinds of daily activities: leaving house, using toilet, taking shower, going to bed, preparing breakfast, preparing dinner, getting drink, in an apartment from 25 Feb 2008 to 21 Mar 2008, as shown in Figure.14.

Figure.14.The Floor Plan of the Test Apartment

This annotated real world datasets contains 245 activities and corresponding state of binary input, as shown in Figure 15.



(a)The State of 14 Binary Input Sensors



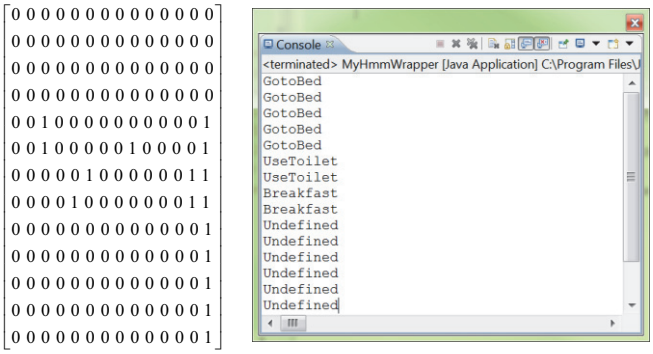
(b)User's Activities

Figure.15.Domestic Activity Dataset

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

The application takes 14 binary input sensors and infers which activity (out of seven) the user is performing. We put our effort only on verifying HMM reasoner, thus sensors information is simulated in this example. We simulate a test sensor data sequence, which is represented by a 15×14 matrix. The row represents the situation of 14 sensors, and the column represents the time slice. The data sequence is put in context-aware middleware successively. Then the learned HMM (parameters determined) reasoner makes inference of the activity sequence by calculating its

probability given an observation sequence (sensor data sequence). The Viterbi algorithm (Viterbi 1967) is used to infer. All the process on HMM is implemented based on jhmm in Java. To make it easier understood, the final result is as shown in Figure.16. As mentioned above, HMM ruler should work all the time. These results and historical results are stored in the context knowledge base, which can be used by various context-aware applications via context-aware middleware.



(a)The simulated input data *(b)The printed results*

Figure.16. The Results of HMM for Domestic Activity.

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 of context awareness. In this work, context information refers to primarily location. From then on, there have been numerous infrastructures to provide service for handling context. The context inference part always plays an important role in these systems. Context toolkit, developed by (Dey, Abowd, and Salber 2001), is a toolkit to support the development of context-aware applications. The part of context interpreter is used to deduce the high-level context information from the low-level information. CASS (Fahy and Clarke 2004) support to context-aware application on hand-held computing devices, and other small mobile computing devices. The important feature of CASS is that it supports to abstract the high-level context and separate context based on inferences and behaviors from the application code. CoBrA (Chen, Finin, and Joshi 2003) is one of earliest systems using semantic web technology to support context-aware pervasive computing. The inference engine of CoBrA is used in the two types of reasoning. Besides detecting and resolving the inconsistent knowledge, it can also infer context knowledge that cannot be easily acquired from the physical sensors. Context reasoner in SOCAM (Gu, Pung, and Zhang 2004) supports two kinds of reasoning: ontology reasoning and user-defined rule-based reasoning. Truong and Dustdar have summarized inference techniques support in existing system (Truong and Dustdar 2009). However they found that context inference is not well adequate and most systems are just based on semantic reasoning.

With emergence of various position sensors, “location” is no longer the researchers’ most concerned context, gradually replaced by "activity". Wang et al. leverage rules based on first-order logic to infer user’s activity (Wang et al. 2004). To better recognize user’s activity, lots of artificial intelligence methods are used in context-aware computing. Panu Korpipaa et al. use a naïve Bayes classifier to recognizer higher-level contexts from lower-level contexts (Korpipaa et al. 2003); Hidden Markov Model is employed to recognize activities in work of (van Kasteren et al. 2008). Pollack et al. use decision tree to make decisions about whether and when it is most appropriate to issue reminder for prescribed activities (Pollack et al. 2003). However these approaches only provide a possibility to solve part of problem in activity context recognition, and none of them take into account how to integrate these approaches in context-aware system and work well together with other parts of system.

Conclusion

In recent work, increasing research work revolves around the “activity” context instead of “location” context. We reviewed literature on activity context recognition in three premier conferences in context awareness during past ten years, summarized all the methods, and concluded the research on activity context recognition into three main facets: basic activity inference, dynamic activity analysis and future activity recommendation.

Based on our previous work, we proposed an intelligent inference engine for context-aware middleware which is made up of a basic inference module and an intelligent inference module. Beside satisfying requirements of checking the context consistency, our inference engine integrates the three most popular methods on activity context recognition: Rules, Decision Tree, and Hide Markov Model. It 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 bus stop scenario, we extract a user’s activity context from his social networks as training set under user’s permission to investigate Rules reasoner and DT reasoner. Concerning domestic activity, we adopt the dataset of domestic activity to verify HMM reasoner. However, the intelligent inference didn’t take into consideration the context

ambiguity problem that contexts from the sensors are not always correct. The context inference engine will reach the conclusion based on inaccurate information. It will lead to take incorrect actions of application, especially in Rules reasoner. We will make improvement of the context inference on this field to enable our system more robust and intelligent.

References

Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P. 1999. Towards a Better Understanding of Context and Context-Awareness. Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing, 304–307. Springer-Verlag, London, UK.

Baum, L., Petrie, T., Soules, G., Weiss, N. 1970. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. The Annals of Mathematical Statistics 41 (1): 164–171.

Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D. 2010. A Survey of Context Modelling and Reasoning Techniques. Pervasive and Mobile Computing 6 (2): 161–180.

Chen, H., Finin, T., Joshi, A. 2003. An Intelligent Broker for Context-Aware Systems. Adjunct Proceedings of Ubicomp: 12–15.

David, B.T., Chalon, R. 2007. IMERA: Experimentation Platform for Computer Augmented Environment for Mobile Actors. WIMOB '07. Washington, DC, USA: IEEE Computer Society.

David, B.T., Zhou, Y., Xu, T., Chalon, R. 2011. Mobile User Interfaces and their Utilization in a Smart City. In The 2011 International Conference on Internet Computing as part of WorldComp 2011 Conference, CSREA Press, 383–388.

Dey, A.K., Abowd, G.D., Salber, D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-aware Applications. Hum Comput. Interact. 16 (2): 97–166.

Fahy, P., Clarke, S. 2004. CASS – a Middleware for Mobile Context-aware Applications. In Workshop on Context Awareness, MobiSys.

Geib, C.W., Maraist, J., Goldman, R.P. 2008. A New Probabilistic Plan Recognition Algorithm Based on String Rewriting. In ICAPS, 91–98.

Gu, T., Pung, H.K., Zhang, D.Q. 2004. A Middleware for Building Context-aware Mobile Services. In Vehicular Technology Conference, 2656–2660.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H. 2009. The WEKA Data Mining Software: An Update. SIGKDD Explor. Newsl. 11 (1): 10–18.

Hu, D.H., Zheng, V.W., Yang, Q. 2011. Cross-domain Activity Recognition via Transfer Learning. Pervasive Mob. Comput. 7 (3): 344–358.

Korpipaa, P., Mantyjarvi, J., Kela, J., Keranen, H., Malm, E.J. 2003. Managing Context Information in Mobile Devices. IEEE Pervasive Computing 2 (3): 42–51.

Krumm, J. ed. 2009. Ubiquitous Computing Fundamentals. Chapman and Hall/CRC.

Lim, B.Y., Dey, A.K. 2010. Toolkit to Support Intelligibility in Context-aware Applications. In Proceedings of the 12th ACM International Conference on Ubiquitous Computing, 13–22. ACM, New York, USA.

Lin, L. 2006. Location-based Activity Recognition . PhD Thesis, University of Washington.

Pazzani, M., Muramatsu, J., Billsus, D. 1996. Syskill & Webert: Identifying Interesting Web Sites. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 54–61. AAAI'96. AAAI Press.

Pollack, M.E., Brown, L., Colbry, D., McCarthy, C.E., Orosz, C., Peintner, B., Ramakrishnan, S., Tsamardinos, I. 2003. Autominder: An Intelligent Cognitive Orthotic System for People with Memory Impairment. Robotics and Autonomous Systems 44(3–4): 273–282.

Rabiner, L. 1989. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77 (2): 257–286.

Romero, D., Parra, C., Seinturier, L., Duchien, L., Casallas, R. 2008. An SCA-Based Middleware Platform for Mobile Devices. In Enterprise Distributed Object Computing Conference Workshops, 2008 12th, 393–396.

Schmidt, A., Beigl, M., Gellersen, H. 1998. There Is More to Context Than Location. Computers and Graphics 23: 893–901.

Truong, H.L., Dustdar, S. 2009. A Survey on Context-aware Web Service Systems. International Journal of Web Information Systems 5 (1): 5–31.

Van Kasteren, T., Noulas, A., Englebienne, G., Kröse, B. 2008. Accurate Activity Recognition in a Home Setting. Proceedings of the 10th international conference on Ubiquitous computing, 1–9.

Vail, D.L., Veloso, M.M., Lafferty, J.D. 2007. Conditional Random Fields for Activity Recognition. Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems: 235:1–235:8. ACM, New York, USA.

Viterbi, A.J. 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. IEEE Transactions on Information Theory 13 (2): 260–269.

Wang, X.H., Zhang, D.Q., Gu, T., 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. 18–23. IEEE Computer Society, Washington, DC, USA

Want, R., Hopper, A., Falcão, V., Gibbons, J. 1992. The Active Badge Location System. ACM Trans. Inf. Syst. 10 (1): 91–102.

Xu, T., David, B., Chalon, R., Zhou, Y. 2011. A Context-aware Middleware for Ambient Intelligence. In Proceedings of the Workshop on Posters and Demos Track, 10:1–10:2. PDT '11. ACM, New York, USA.

Xu, T., Jin, H., David, B., Chalon, R., Zhou, Y. 2013. A Context-aware Middleware for Interaction Device Deployment in AmI. In HCII'13. Las Vegas, USA.