Multiple User Activities Recognition in Smart Home

YaHua Lee, Fuchun Joseph Lin(✉), and Wei-Han Chen

Department of Computer Science,
National Chiao Tung University, Hsinchu, Taiwan
{yahua.cs05g, fjlin, alwayschocol213}@g2.nctu.edu.tw

Abstract. In this paper, we investigate the problem of recognizing multiuser activities using wearable devices in a home environment. Our research objective is to provide situation awareness so that a smart home can respond to the needs of its residents based on the accurate detection of their activities. In this research, we compare applying artificial neural network, decision tree and simple logistic regression for model construction and activity detection. Moreover, we also evaluate different architectural alternatives of our smart home system in order to discover the best system configuration. Our unique contribution lies on the low cost of the proposed system design.

Keywords: Wearable device · Internet of Things · Multiple user activities Artificial neural network · Decision tree · Simple logistic regression

1 Introduction

With proliferation of the IoT technology, our lives are now closely related to smart devices. Many of these devices such as smart phones and wearable devices greatly improve the quality of our life. In particular, wearable devices are becoming more and more ubiquitous. The advances of wearable sensors and wireless networks enable the recognition of human activities based on the continuous readings from various sensors.

Our research objective is to exercise better control in a smart home environment by recognizing multiple user activities based only on data from BLE wearable sensors. We assume these multiple user activities occur in a living room of the smart home environment where the user may watch TV, read newspaper, take a nap on the sofa and do exercising.

We have developed a prototype system based on our research results. In our system, each user is only required to wear a wearable device equipped with accelerometer and gyroscope on his/her waist for activity detection. This wearable device will collect the data from the accelerometer and gyroscope and transmit the data collected to our system via a BLE-paired smartphone. Our system will then derive situation awareness in a living room environment by consolidating multiple users' activities thus recognized. To evaluate various design options for our prototype system, four different activity recognition methods and three different architectural configurations are explored in our experiments.

The rest of the paper is organized as follows: Sect. 2 surveys related work for activity recognition algorithms. Section 3 introduces our system architecture including

Y.-B. Lin et al. (Eds.): IoTaaS 2017, LNICST 246, pp. 202–209, 2018.
https://doi.org/10.1007/978-3-030-00410-1_24

both its hardware and software components. Sections 4 and 5 respectively present our experimental results under different detection methods and architectural configurations. Finally, in Sect. 6 we provide our conclusion and future work.

2 Related Work

To accomplish activity recognition, we adopted several commonly used methods including simple logistic, artificial neural network (ANN) and C4.5 decision tree methods.

- Simple Logistic concerns two-dimensional sample points with one independent variable and one dependent variable (conventionally, the x and y coordinates in a Cartesian coordinate system) and finds a linear function (a non-vertical straight line) that, as accurately as possible, predicts the dependent variable values as a function of the independent variables [1, 2].
- Artificial Neural Network is based on a large collection of neural units, loosely modeling the way a biological brain solves problems with large clusters of biological neurons connected by axons [3, 4].
- C4.5 is an algorithm used to generate a decision tree [5]. This process can be repeated on each derived subset in a recursive manner called recursive partitioning. The attribute with the highest normalized information gain is chosen to make the decision.

Our previous research proposed several methods for the single user activity recognition based on the wearable device [6] including methods of decision tree, hidden Markov model and Viterbi algorithm. These methods also used the assistance of location data derived from Beacons. Our previous research also compared several different architectural configurations for performance evaluation. However, in order to realize a smart home environment, the previous effort on the single user activity recognition is not enough.

Consequently, there have been research efforts in multiple user activity recognition. For example, Rong et al. [7] defined the "combined label states" during model construction with the help of data association. Liang et al. [8] developed a multi-modal, wearable sensor platform to collect sensor data from multiple users, then used Coupled Hidden Markov Model (CHMM) and Factorial Conditional Random Field (FCRF) to model interacting processes in a sensor-based, multiuser scenario. Markus et al. [9] used the ARAS dataset which is a real-world multi-resident dataset stemming from two houses in order to evaluate the proposed algorithm.

In addition to related work mentioned above, we also surveyed other research efforts in activity identification. Overall, the accuracy of activity recognition reported so far is about 83%.

3 System Architecture and Its Components

We explain sensors, processing hardware and mining tools required by our system in the following.

A. Wearable Device

We used a wearable device "Koala" designed by our project team to construct our system. Koala is equipped with two sensors: a 3-axis accelerometer and a 3-axis gyroscope. It uses Bluetooth Low Energy (BLE) as the wireless communication protocol. The specification of Koala is shown in Fig. 1.

The smart phone can establish connection with Koala and receive approximately 30 sets of acceleration and gyroscope data per second. We set the window size used for activity recognition at 60 sets of data and the window movement at every 15 sets of data due to the frequency of motion data.

Whenever the window is full, the system will execute the activity recognition algorithm to identify the user's current activity, then move the window to collect the next 15 sets of motion data. Under this experiment, Koala is worn by a user at his/her waist as depicted in Fig. 2.

B. Smartphone

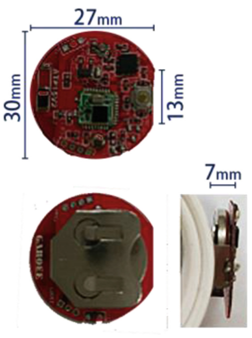
In our system, the smartphone acts as a gateway between Koala and the backend server. We used a Samsung Note 3 with Android 5.0 as the gateway. It has an Octa-core CPU (4 × 1.9 GHz Cortex-A15 & 4 × 1.3 GHz Cortex-A7) and 3 GB of RAM. Samsung Note 3 uses BLE to communicate with Koala and Wi-Fi to communicate with the M2M server such as Python or oneM2M server. Sensors on the phone include Compass Magnetometer, Proximity sensor, Accelerometer, Ambient light sensor, Gyroscope and Barometer.

C. Backend Server

Both Python and oneM2M servers reside in a Raspberry Pi 3 running Raspbian with 1.2 GHz 64-bit quad-core ARMv8 CPU and 1 GB RAM. We developed the former from scratch while adopted OpenMTC from FOKUS for the latter.

D. Smart Home

We use Philips Hue [10] smart lights as the representative appliances in our smart home environment. As depicted in Fig. 3, the M2M server such as Python Server can send RESTful control signals to a Hue bridge in order to control a Hue bulb through ZigBee.



MCU	nRF51822 (Cortex-M0) 16MHz
RAM	16KB
ROM	256KB
Bluetooth	Bluetooth Low Energy (Bluetooth Smart)
Sensor	MPU6050 - 3 axis Accelerometer - 3 axis Gyroscope
LED	Red (Status Indicator)

Fig. 1. Koala hardware and specification

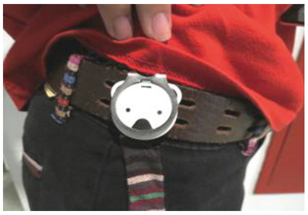


Fig. 2. Koala worn on the waist

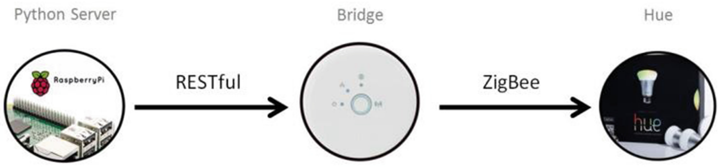


Fig. 3. Hue control flow with Python server

E. Mining Tool

We utilized Weka (Waikato Environment for Knowledge Analysis) as our data mining and machine learning software. Weka supports many machine learning algorithms including artificial neural network, decision tree and simple logistic regression [11].

4 Experimental Results Under Different Detection Methods

We compare the performance of applying artificial neural network, decision tree and simple logistic regression for model construction and activity detection.

After collecting the input data from 10 people, we analyzed 17580 sets of data and derived 293 features based on a window size of 60. These features are used to train and construct the models for activity detection. The IPO diagram is shown in Fig. 4. The dataset was collected from Koala, then via feature extraction was further processed into mean, maximum, minimum and standard deviation for use in activity recognition.

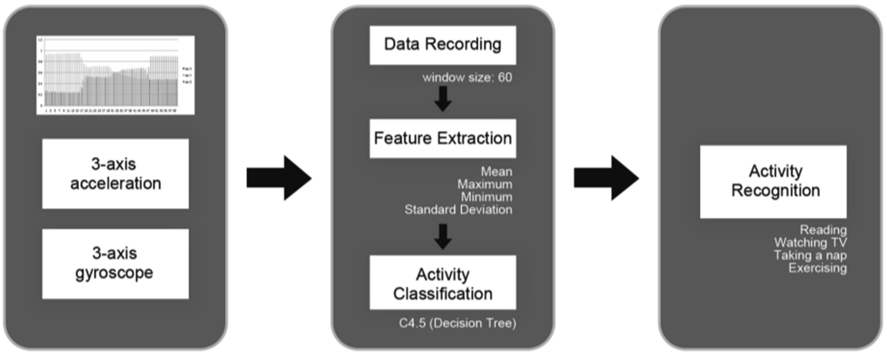


Fig. 4. Input-Process-Output (IPO) diagram

Table 1. Precision results of different algorithms

	Proportion of datasets	Simple logistic	Artificial neural network	Decision tree
Reading	0.300	0.667	0.798	0.944
Watching TV	0.283	0.696	0.871	0.928
Taking a nap	0.185	0.964	0.964	0.981
Exercising	0.232	1.000	1.000	1.000
Weighted avg.		0.807	0.896	0.959

Table 1 shows the precision results of three different algorithms we have applied in activity detection. It shows that Decision Tree always performs the best among three algorithms. In addition, it also shows that the activity "reading" gets the lowest precision due to its confusing motion with activity "watching", while the activity "exercising" gets the highest precision of 100% due to its unique distinguishable features. Since the size of each activity dataset is different, the weighted average is calculated according to the proportional size of each collected activity dataset. Overall, all algorithms could achieve at least 80% of the weighted average precision.

5 Experimental Results Under Different Architectural Configurations

We also evaluate four different architectural alternatives of the system design in order to find out the best system configuration that includes wearable devices, smartphone gateway and backend server.

These four system architectural alternatives are depicted in Figs. 5 and 6. Their differences lie in where the feature processing and activity detection procedures are carried out and what server is used to process the conflict resolution algorithms. The backend is a Python server in both Alternatives 1 and 2; in Alternative 1, the smartphone won't do feature processing but just relay the raw data to the server; on the other hand, in Alternative 2 feature processing is done on the smartphone. Likewise, these are also the differences between alternatives 3 and 4 except now the server is an oneM2M server instead of a Python server.

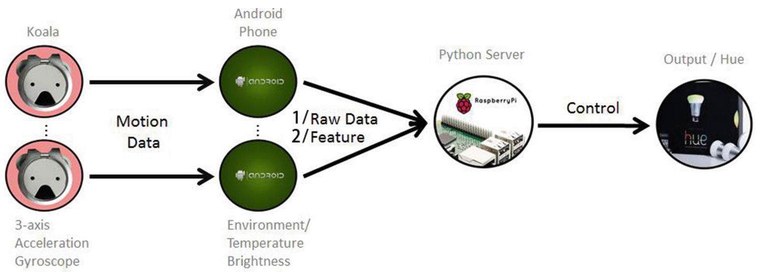


Fig. 5. System architectural alternatives 1 and 2 with Python server

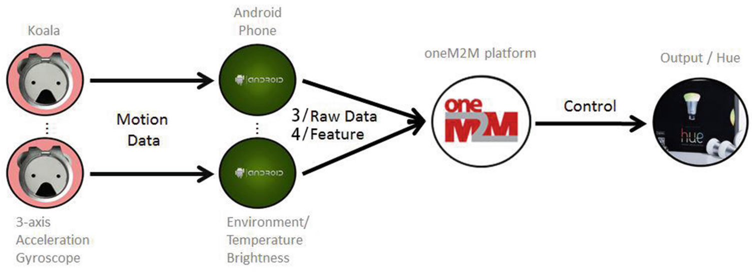


Fig. 6. System architectural alternatives 3 and 4 with oneM2M Server

Alternative 1. No processing by smart phone with Python server as backend
In our first architecture, we send the raw data from the smart phone to the Python server. This means that the Python server will do all the processing including extracting features from these 3-axis acceleration and gyroscope data and carrying out the activity detection and the conflict resolution algorithm.

Alternative 2. Feature processing by smartphone with Python server as backend
In the second architecture, the smartphone will perform feature extraction on the raw data collected from the wearable device with a frequency of once every 2–3 s. The extracted features then will be sent to the Python server for activity recognition and conflict resolution.

Alternative 3. No processing by smartphone with oneM2M server as backend
In the third architecture, we replace the Python server with an oneM2M server. This means the communications between the smart phone and the oneM2M server will follow the oneM2M RESTful standards to post and get the data to and from the server. Feature processing, activity detection and conflict resolution will all be performed on the oneM2M server.

Alternative 4. Feature processing by smartphone with oneM2M server as backend
In the fourth architecture, feature processing will be done in the smart phone. The oneM2M platform only needs to store the features from the RESTful posts and carry out both activity detection and conflict resolution algorithms based on the received features.

To discover which architecture can perform the best, these four architectural alternatives are evaluated based on a set of metrics including memory usage, processing time and CPU utilization.

- **Memory Usage**

The measurement results of memory usage for four architectural alternatives are shown in Fig. 7. Both architectures based on Python server consume a similar size of memory at 1%. On the other hand, the two based on oneM2M demands much more memory resource, especially the architecture starting from raw data; it has 2.8% of memory usage as compared to 1%.

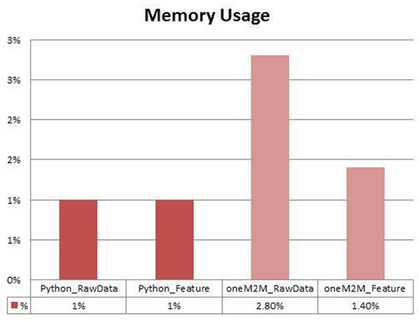


Fig. 7. Memory usage percentage

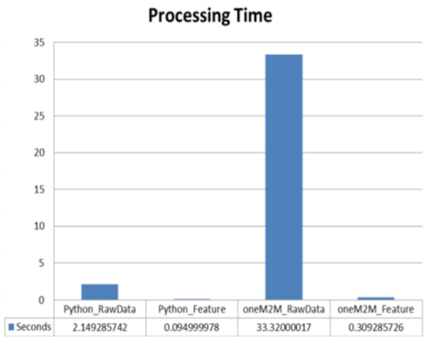


Fig. 8. Processing time result

- **Processing Time**

The processing time results are shown in Fig. 8. It indicates that both architectures starting from raw data will take more processing time than the others starting from feature data. We can reach around 22 times faster in the case of Python server and 107 times faster in the case of oneM2M server. On the other hand, between the Python server and the oneM2M server the former can achieve 3 times faster than the latter; especially in the case of the server starting from raw data, the Python server can obtain 15.5 times faster than the oneM2M server.

- **CPU Utilization**

The CPU utilization is affected by the efficiency of both architecture and algorithm. Here we focus only on the perspective of architecture so all our experiments are based on Decision Tree algorithms. Figure 9 illustrates the CPU utilization results of four architectural alternatives. It indicates the same rising and falling trend in both Python Raw Data and Python Feature systems with a higher CPU utilization by the former. Since processing feature from raw data takes 2 to 3 s, the highest point of CPU utilization appears on the basis of this frequency. It also shows that the oneM2M systems have to spend much more CPU cycles due to the effort on storing and getting data through the platform once every 2 to 3 s. The high rising points of CPU utilization follow this frequency very well. However, in the case of the oneM2M Raw Data system, the CPU utilization not only exhibits an irregular pattern but also stays very busy due to the processing of raw data.

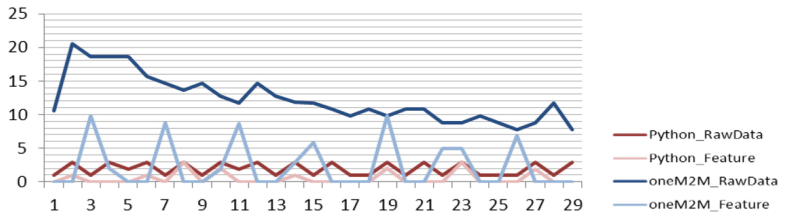


Fig. 9. CPU usage result of different architectures

In Raw Data systems, the CPU utilization of oneM2M (11.997 s) costs 6.48 times more than that of Python server (1.85 s). On the other hand, in Feature systems the loading in oneM2M (2.51 s) is 5.12 times heavier than that of Python server (0.49 s). Also, it is worth noting that the CPU utilization of the Feature systems drops to 0% during a large portion of the time due to the load sharing by the smart phone for feature processing.

As there will be a large number of devices connected to the IoT networks, it is better that work load is distributed among different network components. Therefore, we believe that Feature systems will provide better performance than Raw data systems.

6 Conclusions and Future Work

In this research, we have developed a situation awareness system for smart home based on multiple user activities recognition with four different architectures. Our activity recognition is based on three Machine Learning (ML) algorithms: Simple Logistic, Artificial Neural Network and Decision Tree.

Among three ML algorithms, we conclude that the Decision Tree algorithm at up to 96% of accuracy is the best algorithm for activity recognition. Moreover, among four architectural alternatives, using Python Server with feature processing done on the smartphone will provide a better average performance than the others.

Our unique contribution lies on providing a low-cost solution to activity recognition based on a simple wearable device Koala and a smart phone. Via detecting multiple user activities at home, smart home can be enabled with situation awareness and provide a more convenient and comfortable living environment for home users through the control of lighting, temperature and music. In the future, we plan to address the scalability issues of python and oneM2M servers. We expect better scalability can be achieved by oneM2M server than by Python server when there are numerous devices.

Acknowledgment. The research in this paper is funded by Ministry of Science and Technology (MOST) of Taiwan Government under Project Number MOST 105-2218-E-009-004.

References

1. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **28**(2), 337–407 (2000)
2. Simple linear regression. https://en.wikipedia.org/wiki/Simple_linear_regression. Accessed Feb 2017
3. Bataineh, M.H.: Artificial neural network for studying human performance. MS (Master of Science) thesis, University of Iowa (2012)
4. Artificial neural network. https://en.wikipedia.org/wiki/C4.5_algorithm. Accessed Feb 2017
5. Decision Tree C4.5. https://en.wikipedia.org/wiki/C4.5_algorithm. Accessed Feb 2017
6. Lee, S.-Y., Lin, F.J.: Situation awareness in a smart home environment. In: IEEE WF-IoT 2016, Reston, Virginia, USA, 12/12-14/2016
7. Chen, R., Tong, Y.: A Two-stage method for solving multi-resident activity recognition in smart environments. College of Information Science and Technology, Dalian Maritime University, Dalian, China (2014)
8. Wanga, L., Tao, G., Tao, X., Chen, H., Jian, L.: Recognizing multi-user activities using wearable sensors in a smart home. Pervasive Mobile Comput. **7**, 287–298 (2011)
9. Prossegger, M., Bouchachia, A.: Multi-resident activity recognition using incremental decision trees. In: Bouchachia, A. (ed.) ICAIS 2014. LNCS (LNAI), vol. 8779, pp. 182–191. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11298-5_19
10. Hue, P.: Philips. http://www2.meethue.com/. Accessed Feb 2017
11. Weka (machine learning), January 2017. https://en.wikipedia.org/wiki/Weka_(machine_learning). Accessed Feb 2017