Design and Implementation of a Context-Aware Mobile Data Automatic Collection System (Post-print)

Authors: Xia Lixin, Yang Jinqing, Cheng Xiufeng

Date: 2017-11-08T00:00:00+00:00

Abstract

Objective: To provide a design framework based on context awareness technology for data collection and analysis in mobile environments, thereby optimizing the automated mobile data collection process.

Background: Although context-aware data collection based on mobile devices has advanced beyond manual and semi-automatic methods in traditional network environments, applications that leverage underlying mobile sensors to directly acquire real-time user information, achieve real-time, dynamic, and comprehensive data collection and mining, and consequently deliver proactive services remain in a developmental stage.

Methods: By utilizing the numerous built-in sensors of the Android platform, we designed a data collection framework featuring automatic client-side acquisition and proactive server-side reception. For empirical evaluation, we designed a prototype that recommends high-level services through dynamic context awareness.

Results: Empirical results demonstrate that the system can effectively collect mobile user data for specific contexts, providing robust support for high-level recommendation services.

Limitations: The collected data exhibits significant redundancy, and multi-perspective, comprehensive, and in-depth context reasoning has not yet been implemented, necessitating more thorough analysis of user data in future research.

Conclusion: The context-aware mobile data automatic collection system enables proactive collection of mobile data and facilitates push services to individuals or groups based on collection results, thereby serving as a valuable support for both user behavior research and context computing studies in mobile environments.

Full Text

Preamble

ChinaXiv Partner Journal: Design and Implementation of a Mobile Data Automatic Collection System Based on Context Awareness Technology

Xia Lixin, Yang Jinqing, Cheng Xiufeng
(School of Information Management, Central China Normal University, Wuhan 430079, China)

Abstract

[**Objective**] This paper proposes a design framework based on context awareness technology for mobile data collection and analysis, aiming to optimize the automated mobile data collection process. [**Application Background**] Although context-aware data collection on mobile devices has advanced beyond traditional manual and semi-automatic methods in network environments, applications that leverage mobile 底层 sensors to directly obtain real-time user information for real-time, dynamic, and comprehensive data collection and mining —ultimately achieving proactive service delivery—remain in the developmental stage. [**Methods**] Utilizing the numerous built-in sensors of Android, we designed a data collection framework featuring automatic client-side acquisition and active server-side reception. For empirical evaluation, we implemented a case study demonstrating dynamic context-aware recommendation of high-level services. [**Results**] Empirical results show that the system can effectively collect mobile user data for specific contexts, providing robust support for high-level recommendation services. [**Limitations**] The collected data exhibits significant redundancy, and multi-angle, comprehensive, and in-depth context reasoning has not yet been performed, requiring deeper user data analysis in future research. [**Conclusions**] The context-aware mobile data automatic collection system enables proactive mobile data collection and pushes services to individuals or groups based on collection results, serving as valuable support for both mobile user behavior research and context computing studies.

Keywords: Context Awareness; Application Framework; Android; Context Reasoning; User Behavior
Classification Numbers: G25; TP311

1. Introduction

The rapid development of mobile communication technology has led to the widespread adoption of mobile intelligent terminals. Smartphones have shortened communication distances, improved work efficiency, and facilitated social life. On one hand, this popularity places higher demands on researchers and service providers seeking to obtain user data; on the other hand, smartphones

contain vast amounts of authentic behavioral data, making the benefits of using smartphone data for context computing and related behavior research self-evident. If we can quickly acquire this data, mine effective information, perceive user environments, and thereby deduce user needs to proactively provide corresponding services, we can significantly enhance the reliability and accuracy of behavioral and service research. Meanwhile, in the field of contextual computing [1], researchers select various context information according to their needs and employ multiple methods—including IoT technology, cloud computing, backend databases, network storage, and sensor technologies—to monitor and obtain environmental information, regularizing these typically massive and complex user information sets. The regularization process specifically involves: first establishing a formal model (static), then setting matching rules for the static model to dynamically derive user needs, filtering the most suitable user services, and thereby performing service recommendations [2].

The above describes the general process of context computing. Overall, it includes data collection and standardization. For the latter, selecting appropriate models, rule languages, middleware development, and service matching requires finding a methodological system applicable to specific environments and context-aware systems to unify existing data and models. For the former, the quality of acquired data—including precision, accuracy, and uniformity—directly impacts subsequent context computing processes. However, mobile data collection faces various technical bottlenecks and security/privacy issues, causing much research based on mobile data collection to remain at the theoretical model stage. Consequently, how to efficiently and automatically collect mobile context data, and further design a robust context computing framework (including context modeling, context reasoning, aware services, and middleware technology) that enables systems to perceive user states and enhance user experience [3], constitutes a primary focus of context computing research.

Context information acquisition targets diverse object types, with corresponding acquisition methods equally varied. Physical information such as spatial location, time, network, Bluetooth, battery level, temperature, humidity, light, and noise is generally obtained through physical sensors onboard devices or by using multiple communication protocols to unify device interfaces [4]. High-level contexts like nearby objects, traffic, crowds, customs, and scenery require derivation through combinations of primary context sensing. To identify high-level contexts from primary sensor data, researchers typically focus on task execution scenarios in user activities according to research objectives, conducting investigations, decomposing tasks, and computing contexts. Additionally, combining data mining and visualization technologies to discover implicit relationships among context information and visualize them represents another development direction for context awareness technology.

Android smart mobile terminals (version 4.0 and above) embed multiple sensors capable of collecting large amounts of accurate, real-time, and reliable data, making context-aware data collection from terminals feasible [5]. According to

pervasive computing [6] and mobile computing theories [7], data collected from physical sensors can be transformed into context information shared among multiple users and tasks. Building upon the mobile sensing software AWARE [8], we developed a context-aware system suitable for specific context analysis (abbreviated as "MDCF") and designed a relatively simple yet easily extensible context standardization framework to enable better perception of context and user behavior from data collected by this system. MDCF can abstract raw information and low-level context information into high-level context information understandable by ordinary applications, which is then provided to relevant applications. The general process involves: collecting raw data through phone sensors, abstracting the data into context information, and finally storing context information on SD cards or remotely transmitting it to server-side storage, thereby achieving mobile data collection, management, utilization, and research.

Context computing involves broad domains, characterized by complex system architecture hierarchies and diverse technologies, including context data acquisition, formal model establishment, and data analysis and processing. Since Schilit et al. proposed the concept of context computing in 1994 [9], numerous scholars have studied context-aware information using semantic web, IoT technology, ontology theory, cloud computing, complex networks, SOA, and many other methods. Focusing on context data acquisition technology—the emphasis of this paper—domestic and international scholars have conducted substantial research. For example, Device Information Access (DIA) middleware shields 底层 heterogeneous data through unified interfaces [10], while Xu et al.' s Cabot component addresses consistency issues in dynamic monitoring data [11]. Han Li et al. extracted user experience data through comprehensive experimental methods including questionnaires, screen recording, and audio recording [12]. Additionally, in specific domains, multiple methods have been applied to collect data for industry services [13-14].

However, the concept of context-aware systems has never been precisely defined. Generally, context-aware systems are considered to use context data to provide relevant information and services to users, with their relationships depending on user tasks [15]. Nevertheless, context-aware technology has received significant attention from industry, yielding numerous application achievements. CORTEX [16] integrates mobile smartphone sensor data, recording and reasoning context data through mobile data collection software. It uses an event communication protocol to exchange data among various sensors, actuators, and applications, employing event-condition-action rules to infer new context information. Context Studio [17] applies user mediation and explainability to context reasoning, allowing users to control context data through input information and using the Blackboard method to combine multiple contexts to generate new high-level contexts. ContextPhone [18] treats collected information and users themselves as resources, understanding and controlling context information exchange according to user comprehensibility, and supporting communication between applications and users. AWARENESS [19] emphasizes user privacy issues and applies environmental quality concepts to express context information qual-

ity characteristics, solving user privacy problems by controlling shared context information. Momento [20] collects qualitative and quantitative data through multiple communication methods, generating multiple types of context information with remote control capabilities. MyExperience [21] collects various sensor-based and user-based data, supporting customizable, user-based qualitative data collection and random synchronous uploading of data to servers. CenceMe [22] infers individual perceived presence states by leveraging mobile smartphone sensors and information shared through social networking applications, extracting user life patterns and habits. It supports offloading processing to devices with asynchronous data mining and high processing capabilities. EmotionSense [23] senses personal emotions, behavioral activities, language, and close-range interactions among friends, supporting online, offline, and multidisciplinary scalable context information. SystemSens [24] collects data and information on user interactions with application software, with its built-in debugging functions generating information that supports not only software operation but also context reasoning and data collection. AWARE [25] collects various sensor data, abstracts it to generate high-level context information, and achieves visualization and remote transmission storage. Compared with similar open-source systems, AWARE' s advantages lie in plugin extensibility and reusability.

3. Requirements and Technical Approach

Currently, approaches for collecting mobile user behavior data for scientific research can be summarized into three categories: manual collection, shared collection, and active collection. Manual collection refers to researchers directly downloading, copying, or requesting data from data owners' servers according to experimental requirements. Shared collection involves obtaining required research data through third-party plugins using interfaces provided by open platforms. Based on literature we have collected, shared collection is widely used for obtaining research data. However, the interface opening rights for this approach are controlled by operators, creating considerable access barriers for researchers. Active collection systems refer to researchers independently developing data collection software according to experimental needs or selecting open-source software development components that meet research requirements.

The advantages of active collection lie in its autonomy and flexibility. However, it still faces issues of uncontrollability, incompleteness, non-extensibility, and lack of visualization: uncontrollability mainly refers to collection software being unable to selectively obtain required research data due to functional compatibility issues; incompleteness means systems often fail to collect data within the scope required by research needs, resulting in incomplete data; non-extensibility indicates that collected data cannot meet expansion needs due to interface issues.

In light of these problems, we constructed the system by combining Android' s built-in data sharing standard technology, visualization technology, data mining, and machine learning technology from the perspective of research data require-

ments. System objectives mainly include "sensor data transmission," "context standardization and combination," and "service discovery." The technical approach is illustrated in Figure 1 [Figure 1: see original paper].

Figure 1. Overall Technical Approach of the Mobile Data Automatic Collection System Based on Context Awareness

(1) **Collection Layer Function: Sensor Data Transmission.** MDCF controls the collection and remote storage of sensor data from phone sensors through programming, using the getDefaultSensor() method to instantiate each sensor and assigning a start button to each sensor in the sensor list on the interactive interface for selective context collection.

(2) **Standardization Layer Primary Function:** Based on shared data obtained through Android data sharing standards, we perform context modeling for research objects. First, we establish a metadata representation model (ontology), using standard ontology specifications to infer associations between ontologies and among tuples, thereby acquiring user primary (static) context.

(3) **Standardization Layer Advanced Function:** From static contexts, we use description logic for high-level reasoning, generating dynamic context oriented toward services for different contexts.

(4) **Service Layer Function:** We provide services based on high-level contexts and feed the results back into MDCF for further service discovery.

4. System Design

4.1 System Architecture

As the collection component of a context-aware system, the MDCF framework adopts a standard C/S architecture. It mainly includes a server, server-side database (MySQL), data transmission method (MQTT), mobile phone client (Android), and local lightweight database (SQLite). The MDCF system structure framework is shown in Figure 2 [Figure 2: see original paper].

Figure 2. Mobile Data Collection System Architecture

MDCF consists of server and client components. The server-side encapsulates service request processing and backend database (MySQL) access operations, transmitting local data in JSON object format to the server via the HTTPS protocol. It uses the MQTT protocol to actively and real-time exchange context information with clients. The server receives data transmitted from client applications and manages data stored in the server database, providing data support for scientific research. The client records, collects, and processes sensor data on mobile smartphones through various classes provided by Android development (e.g., Sensor class).

4.2 Logical Architecture

MDCF stores, transmits, mines, and abstracts raw data generated by mobile devices to form primary context information and visualizes some information. Therefore, MDCF is subdivided into five layers: raw data collection layer, data storage layer, data exchange layer, context layer, and visualization layer, as shown in Figure 3 [Figure 3: see original paper].

Figure 3. Mobile Data Collection System Hierarchy Diagram

(1) **Raw Data Layer:** This refers to the first storage medium obtained from various phone sensors. Phone sensors include not only hardware sensors but also software sensors and user-based sensors. This system has 28 hardware sensors, software sensors, and behavior sensors. HW represents hardware sensors that directly obtain parameters from phone hardware (e.g., gyroscope); SW represents sensors that obtain information through software (e.g., APP) information sharing; H represents behavioral information obtained by treating users as sensors (e.g., questionnaires), which cannot be sensed through hardware or software. Some sensors used by MDCF of these three types are shown in Table 1 .

Table 1 . Partial Sensor List

Sensor	Description
Accelerometer	Represents acceleration forces along device axes, including gravity
Application	Records application names running in foreground and background on the device
Barometer	Records atmospheric pressure-related sensor information
Battery	Generates battery and power event data (e.g., restart, shutdown)
Bluetooth	Built-in Bluetooth sensor generates information and performs interval scanning of nearby Bluetooth devices
Communication	For user communication activities, such as call status, message status
ESM	Obtains user-provided data through ESM questionnaires, triggered by context events, time, etc.
Gravity	Represents gravity-related sensor information along device axes
Installations	Records information about applications added, deleted, or updated on mobile devices
Light	Senses ambient brightness information

Sensor	Description
Locations	Records network and GPS location information of mobile devices and provides the most reliable location information
Network	Records network usage, such as mobile networks, WiFi
Orientation	Represents azimuth-related sensor information along device axes
Processor	Records workload of system, user, and mobile processor
Proximity	Records distance between device and objects
Screen	Records screen status and user lock and unlock events

(2) **Data Storage Layer:** Includes local storage and remote storage. Local storage refers to storing context data in the lightweight database SQLite on the phone; remote storage refers to uploading context data to the remote server database via Web services.

(3) **Data Exchange Layer:** Provides technical support for context information sharing and exchange mechanisms for local and external context data. The mobile data collection system uses MQTT technology to support context data exchange between mobile smartphones and servers and other devices. Internally, the MDCF client uses Android' s Broadcast and Observer methods for context data transmission.

(4) **Context Layer:** Uses simple conditional rules to abstract raw data into primary context information. Primary context information can be shared in the exchange layer and can further generate dynamic context information from static context information.

(5) **Visualization Layer:** Uses context plugins to simply visualize context information from the context layer, enabling user interaction with context information. It has multiple functions, such as sharing selection, sensor lists, and current status.

4.3 Client Implementation

(1) **Sensor Control Function**
Sensor control is a key technology of MDCF. Implementing sensor control requires appropriate programming for different sensors and their types: physical sensors on the phone itself use Android' s Sensor class to obtain specified sensor types through the SensorManager' s getDefaultSensor(int TYPE) method; software sensors utilize Android data sharing standards to achieve data exchange between different applications (detailed according to programs); behavior sensors input difficult-to-record data information through user-system interaction, such as pushing questionnaires

via MQTT protocol to remind users to fill them out. Researchers can independently activate personalized sensors according to research needs, implementing personalized control interfaces for various sensors through Android' s PreferenceActivity combined with PreferenceFragment. Based on the above technologies, various sensor instantiations are implemented as shown in Table 2 (where parameter TYPE identifies different sensors, TYPE values are encapsulated in the Sensor class, and -1 represents generic sensor types).

Table 2 . Sensor Class

Sensor Name	Description
TYPE_{ACCELEROMETER}	Accelerometer sensor
TYPE_{GYROSCOPE}	Gyroscope sensor
TYPE_{LIGHT}	Light sensor
TYPE_{{{MAGNETIC}}_{{FIELD}}}	Magnetometer sensor
TYPE_{ORIENTATION}	Orientation sensor
TYPE_{PRESSURE}	Pressure sensor
TYPE_{PROXIMITY}	Proximity sensor
TYPE_{TEMPERATURE}	Temperature sensor
TYPE_{APPLICATION}	Application logging sensor
TYPE_{ALL}	All sensors

(2) **Plugin Management Function**
Considering the diversity of research data collection needs, MDCF reduces system coupling, opens system software interfaces, and improves software application extensibility. Leveraging plugin flexibility enables abstraction of context information, subsequently integrating technologies such as machine learning and data mining algorithms into context reasoning to abstract raw data into higher-level contexts.

(3) **Visualization Function**
Client-side visualization is one of MDCF' s main features. Unlike other one-click collection software, it provides feedback on information collection status, monitoring the entire data collection process to some extent. This not only improves user experience but also ensures trust in human-computer interaction, enabling users to promptly understand and improve transmission strategies, adjust collection scope, and guarantee data standardization for researchers and user privacy security.

4.4 Server Management Implementation

The server side mainly builds a lightweight service based on Tomcat+MySQL, primarily receiving context data uploaded by clients and enabling autonomous management and abstract analysis of collected information. The MDCF server combines Web and MQTT methods to achieve remote storage of context data

and information exchange with clients. The server-side composition is shown in Table 3 .

Table 3 . Server-Side Structure Hierarchy

Data Collection System Server	Description
Web Server	Data MVC Databases
MQTT Server	Messages Repository Messages

Through the Web server and MQTT server, remote data transmission can be configured on the client side. The MQTT server supports distributed infrastructure and real-time context information exchange with user clients. This server is an RSMB publish/subscribe broker that can also be clustered to support load balancing of context information. To ensure security and privacy, RSMB in MQTT supports secure authentication connections. Message storage methods are file-based or database-based, primarily storing context information, events, and data. Web services follow the MVC design pattern, where the controller loads corresponding object models and executes commands based on requests. If visualization is requested, the controller additionally loads corresponding views. Therefore, MDCF services include both Web and MQTT servers, combining their characteristics to improve performance.

4.5 Application Effects

After client startup, the interface shown in Figure 4(a) [Figure 4: see original paper] is displayed first. It shows software and hardware sensor options contained in MDCF and displays client unique identifier information, where the unique identifier is generated when users install the software. The plugin management function can open the add plugin interface, displaying MDCF system-compatible plugins for data collection and visualization, as shown in Figure 4(b) [Figure 4: see original paper]. The raw context status visualization interface can display more context information based on activated sensors and plugins, as shown in Figure 4(c) [Figure 4: see original paper].

5. Empirical Study: APP Usage Analysis

5.1 Context Modeling

Context modeling primarily uses theoretical models or methods to formally represent obtained contexts as meaningful clues. Selecting appropriate formal methods to express and store context information and establish context models is crucial work after context information acquisition. Currently, classic formal methods include colored Petri nets [26], object-oriented methods [27], and ontology models [28]. Colored Petri nets can express limited context information and are only suitable for distributed concurrent processes. Object-oriented models have strong expressive capabilities but lack rule support [1]. Ontology languages

[29] based on ontology theory can establish knowledge representation and reasoning systems with logical description, reasoning, and expressive capabilities, effectively describing context information and achieving unity in formal and semantic reasoning of collection results. This paper selects an ontology-based context model to describe collected context data.

The goal of MDCF system ontology construction is to obtain two levels of context: low-level and high-level. Low-level context-aware modeling is task-derivation type, using context models and data logic to infer users' current tasks (e.g., determining what users are doing based on which APP they are using). High-level context-aware modeling is more service-oriented, mining associations between ontologies and among ontology tuples, performing algorithmic reasoning on each association to infer users' contextual needs (e.g., inferring which mobile phone brands are more likely to have battery issues affecting consumption by checking APP error rates, device temperature differences, and user usage frequency).

We preliminarily define a context information ontology (APP usage ontology) containing a 9-tuple description <Location, Device, Temperature, Battery, App-Foreground, App-History, App-Notification, App-Crash, WIFI> and its internal tuple association relationships. As shown in Table 4 , each tuple contains a series of sensor records (some record names are repeated but do not represent other tuples, as they come from different sensors).

Table 4 . Partial Tuples and Record List for MDCF Context Modeling

Tuple	Description	Fields
Location	Location=<device_{id},timestamp,latitude,longitude,accuracy>	device_{id}, timestamp, latitude, longitude, accuracy
Device	Mobile brand, model, and software version Device=<device_{id},timestamp,hardware,board,product>	device_{id}, timestamp, hardware, product
Temperature	External phone temperature Temperature=<device_{id},timestamp,tempOut, accuracy>	device_{id}, timestamp, tempOut, accuracy
Battery	Phone battery temperature Battery =<device_{id},timestamp,tempIn, battery_{level}>	device_{id}, timestamp, tempIn, battery_{level}
WIFI	Mobile WiFi, WIFI= < device_{id}, timestamp, ssid, frequency >	device_{id}, timestamp, ssid, frequency

Tuple	Description	Fields
App-Foreground	Currently running APP App-Foreground=<device_{id},timestamp,app_{name}, Is_{{{sys}}_{{app}}} >	device_{id}, timestamp, app_{name}, Is_{{{sys}}_{{app}}}
App-History	Mobile APP running history App-History= < device_{id}, timestamp,app_{name}, end_{time} >	device_{id}, timestamp, app_{name}, end_{time}
App-Notification	Mobile application notifications App-Notification= < device_{id}, timestamp,app_{name}, text >	device_{id}, timestamp, app_{name}, text
App-Crash	Mobile application runtime errors App-Crash= < device_{id}, timestamp,app_{name}, error >	device_{id}, timestamp, app_{name}, error

After establishing the ontology model, associations between tuples can be mined to derive high-level context-aware information as defined above. As shown in Figure 5 [Figure 5: see original paper], during mobile phone usage behavior, life patterns (Life) can be inferred based on APP usage patterns, information interference states (Influence) can be inferred based on mobile notification information, user experience information for different brands can be inferred based on internal/external temperature differences, brands, and APP usage information (Temperature), and user acceptance of APPs can be inferred based on APP error rates and APP usage information.

5.2 Context Reasoning

Collecting data through direct context acquisition and establishing context models through formal methods belong to the a priori stage of context computing, with difficulties lying in the selection of unified processing models for data formats [1]. However, direct context cannot describe users' complete current context; reasoning rules are needed to infer middle and high-level context information or hidden information from obtained contexts [30]. Context reasoning methods include: similarity-based computational reasoning [31], multi-valued logic-based reasoning [32], rule-based reasoning [33], ontology-based reasoning, and context reasoning based on ontology and custom rules [34]. Among them, context reasoning based on ontology and custom rules uses custom description logic and ontology models to define contexts, context relationships, and rules as ontologies, ontology properties (tuples), and constraint relationships. Its typical representative, OWL [35], expands model information by executing SWRL rules, suitable for middle and high-level context reasoning. SWRL [36] rules are expressed as follows:

The semantics are: as long as conditions represented in the antecedent are

satisfied in the model, facts described in the consequent must also exist in the model. If facts in the consequent do not exist in the model, the reasoning engine will add them. In this empirical study, we performed context reasoning based on several tuple associations (middle-level context) in the "APP usage" ontology to obtain high-level context information, as shown in Table 5 .

Table 5 . Reasoning Rules from Current Context to High-Level Context in "APP Usage Ontology"

Association (Middle-Level Context)	Reasoning Applicable Algorithm	Description
Life (Life-type context)	K-means, SOM, FCM···	Indicates certain life patterns exist in App-History
Influence (Influence-type context)	SVD, NSFCM, VSM···	Indicates interference of notification information in App-Notification
Experience (Experience-type context)	Apriori, FP Growth, Eclat···	Indicates correlations among Device, Location, and tempIntemp
Mental (Psychology-type context)	Apriori, FP Growth, Eclat···	Indicates correlations among App-Crash, App-History, and Device

As shown in Table 5, the relationship between device information (Device) and application error rate (App-Crash) can determine whether the user' s current phone status is "lagging," suggesting software version upgrades or phone replacement.

Reasoning Rule 1: Reasoning rule based on user APP usage patterns: <timestamp,end_{timestamp}|Life> App-History

From the start and end times of an APP in the App-History tuple, using appropriate clustering algorithms, we can infer usage patterns from current usage contexts.

Reasoning Rule 2: Reasoning rule based on interference of user-received notification information:
<app_{name}, text|Influence> App-Notification
From notification information in the App-Notification tuple, using relevant text mining algorithms to extract keywords, we can determine notification types (interfering/non-interfering) and infer whether intelligent interference message blocking can be performed from current contexts.

Reasoning Rule 3: Reasoning rule based on phone temperature differences at different times:
<temperature, app_{name}|Experience> Device, Battery
From relevant context information in related domains, using correlation algorithms to infer implicit relationships between phone temperature and user experience.

Reasoning Rule 4: Reasoning rule based on temperature differences, usage, and error rates at different times:
<app_{name}, temperature, timestamp|Mental> Device, App-Foreground, App-Crash
From relevant context information, using association algorithms to infer user psychological information.

To verify the applicability of the MDCF collection system and corresponding context reasoning framework, we collected 28 valid individual datasets from September 15, 2016, to November 15, 2016 (30 datasets were actually collected). Learning APP records totaled 104,052 entries, describing 18 APP usage information. We extracted a simple list related to user tasks from different tuples of the previously defined "APP usage" ontology, as shown in Table 6 .

Table 6 . Example of Static Context

Device_{id}	Timestamp	APP_{name}	Battery	Location	Task
a32f534c4a	2016-09-09 15:01:18	Youdao Dictionary	0.98	114.3542, 30.5123	Learning
a32f534c4a	2016-09-09 15:18:00	Youdao Dictionary	0.96	114.3542, 30.5123	Learning
a32f534c4a	2016-09-09 15:19:45	Youdao Dictionary	0.96	114.3542, 30.5123	Learning
a32f534c4a	2016-09-09 16:02:18	Youdao Dictionary	0.94	114.3542, 30.5123	Learning
a32f534c4a	2016-09-09 16:28:22	Youdao Dictionary	0.92	114.3542, 30.5123	Learning

Based on the previously established ontology model, we input discrete raw data into the ontology, extracted associations, and performed context combination: unique identifier, time, location, battery level, and task, processing to obtain static context information. As shown in Table 6, the user used the "Youdao Dictionary" APP at the same location (within a certain teaching building) during a certain time period. Next, we extracted APP-related information from different tuples, with fields shown in Table 7 .

Table 7 . Dataset Format Description

Field	Description
Deviceid	Device ID, used to associate different APP usage within the same phone
APP_{name}	Corresponds to different APPs for different users, composed of "."
timestamp	Time when different users start and terminate APP usage

Applying Reasoning Rule 1 and using simple clustering algorithms to find correlations between an APP and other APPs enables collaborative recommendation services. For example, we defined 7 APP categories: system management, social communication, learning tools, online shopping/payment, search tools, entertainment, and utilities. Figure 6 [Figure 6: see original paper] shows the correlation degree derived from user usage frequency among these 7 APP categories, where nodes represent APPs, colors represent APP types, and An represents clustering results.

From Figure 6, we can find that A2 and A3 have high internal correlation. Although A2 contains fewer APPs, when using learning tools 91 (Mobile Reading) and 51 (Migu Reading), users also tend to use notepad tools. Based on this dynamic context, notepad APPs can be proactively recommended when users download or use reading APPs.

6. Conclusion

This paper designed a mobile data collection, reasoning, and analysis framework based on phone sensors and the AWARE architecture. For APP usage scenarios and custom reasoning rules, we further performed dynamic context reasoning on obtained data to derive user patterns and provide proactive services based on these patterns. Empirical results demonstrate that mobile data collection based on context awareness technology can achieve data standardization, reasoning, and service discovery. This undoubtedly provides support for data collection researchers. Future efforts should focus on localizing and personalizing mobile collection systems according to China' s specific mobile data collection environment, establishing complete context reasoning systems. Additionally, applying

middleware technology and correlation algorithms for high-dimensional, high-level context reasoning and behavior discovery represents an important future direction in context computing.

It is worth noting that data collected by MDCF is suitable for limited reasoning from raw context data to low-level static context and then to high-level dynamic context. However, this is not the ultimate goal of context awareness; the final purpose is service optimization. Service types are diverse, so multiple service systems can be developed for different contexts on this basis. Furthermore, applying middleware technology and correlation algorithms for high-dimensional, high-level context reasoning and behavior discovery is an important future development direction in context computing.

References

[1] Li Weiping, Wang Wusheng, Mo Tong, et al. Survey of Contextual Computing [J]. Journal of Computer Research and Development, 2015, 52(2): 542-552.

[2] Mo Tong, Li Weiping, Wu Zhonghai, et al. Framework of Context-Aware Based Service System[J]. Chinese Journal of Computers, 2010, 33(11): 2084-2092.

[3] Hassenzahl M, Tractinsky N. User Experience - A Research Agenda[J]. Behaviour & Information Technology, 2006, 25(2): 91-97.

[4] Li W, Chu W, Tung F, et al. A Uniform Device Information Access for Context-Aware Middleware[C]//Proceedings of IEEE International Conference on Web Services. IEEE Computer Society, 2010:654-657.

[5] Abowd G D, Dey A K, Brown P J, et al. Towards a Better Understanding of Context and Context-Awareness[C]//Proceedings of International Symposium on Handheld & Ubiquitous Computing, Karlsruhe, Germany. London, UK: Springer-Verlag, 1999: 304-307.

[6] Weiser M. Hot Topics-Ubiquitous Computing[J]. Computer, 1993, 26(10): 71-72.

[7] Imielinski T, Korth H F. Mobile Computing[M]. Boston: Kluwer Academic, 1996.

[8] Android Mobile Context Instrumentation Framework [EB/OL]. [2017-02-10]. http://www.awareframework.com/.

[9] Schilit B, Adams N, Want R. Context-Aware Computing Applications[C]//Proceedings of the Workshop on Mobile Computing Systems & Applications. Washington, USA: IEEE Computer Society, 1994: 85-90.

[10] Li W, Chu W, Tung F, et al. A Uniform Device Information Access for Context-Aware Middleware[C]//Proceedings of the 2010 IEEE International Conference on Web Services. Washington, USA: IEEE Computer Society, 2010:654-657.

[11] Xu C, Cheung S C. Inconsistency Detection and Resolution for Context-Aware Middleware Support[J]. ACM SIGSOFT Software Engineering Notes, 2005, 30(5): 336-345.

[12] Han Li, Liu Zhengjie, Li Hui, et al. A Method Based on Context-Awareness for Remote User Experience Data Capturing [J]. Chinese Journal of Computers, 2015,38(11):2234-2246.

[13] Xue Xiao, Chang Jingkun, Zeng Zhifeng, et al. Context-Aware Intelligent Service System for Mine Industry [J]. Computer Engineering & Science, 2013, 35(9): 36-44.

[14] Jiang Dayang, Yao Qi, Ji Yun. Study on Strategies of Context Awareness Data Collection Based on Smart Campus [J]. Journal of Changzhou Vocational College of Information Technology, 2016, 15(4): 29-31.

[15] Dey A K. Understanding and Using Context[J]. Personal and Ubiquitous Computing, 2001, 5(1): 4-7.

[16] Biegel G, Cahill V. A Framework for Developing Mobile, Context-aware Applications[C]//Proceedings of the 2nd IEEE Conference on Pervasive Computing and Communications. IEEE, 2004: 361-365.

[17] Korpipaa P, Mantyjarvi J, Kela J, et al. Managing Context Information in Mobile Devices[J]. IEEE Pervasive Computing, 2003, 2(3): 42-51.

[18] Raento M, Oulasvirta A, Petit R, et al. ContextPhone: A Prototyping Platform for Context-Aware Mobile Applications[J]. IEEE Pervasive Computing, 2005, 4(2): 51-59.

[19] Wegdam M. AWARENESS: A Project on Context AWARE Mobile NEtworks and ServiceS[J]. Mobile & Wireless Communication Summit, 2005, 6: 19-23.

[20] Carter S, Mankoff J, Heer J. Momento: Support for Situated Ubicomp Experimentation[C]//Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, California, USA. New York, USA: ACM, 2007:125-134.

[21] Froehlich J, Chen M Y, Consolvo S, et al. MyExperience: A System for in Situ Tracing and Capturing of User Feedback on Mobile Phones[C]//Proceedings of the 5th International Conference on Mobile Systems, Applications and Services. New York, USA: ACM, 2007: 57-70.

[22] Miluzzo E, Lane N D, Eisenman S B, et al. CenceMe –Injecting Sensing Presence into Social Networking Applications[C]//Proceedings of EuroSSC 2007. Berlin, Germany: Springer, 2007: 1-28.

[23] Rachuri K K, Musolesi M, Mascolo C, et al. EmotionSense: A Mobile Phones Based Adaptive Platform for Experimental Social Psychology Research[C]//Proceedings of the 12th ACM International Conference on Ubiquitous Computing, Copenhagen, Denmark. New York, USA: ACM, 2010: 281-290.

[24] Falaki H, Mahajan R, Estrin D. SystemSens: A Tool for Monitoring Usage in Smartphone Research Deployments[C]//Proceedings of the 6th International Workshop on MobiArch, Maryland, USA. New York, USA: ACM, 2011: 25-30.

[25] Ferreira D, Kostakos V, Dey A K. AWARE: Mobile Context Instrumentation Framework[J]. Frontiers in ICT, 2015, 2: Article 6.

[26] Kwon O B. Modeling and Generating Context-Aware Agent-Based Applications with Amended Colored Petri Nets[J]. Expert Systems with Applications, 2004, 27(4): 609-621.

[27] Henricksen K, Indulska J, Rakotonirainy A. Modeling Context Information in Pervasive Computing Systems[C]//Proceedings of the 1st International Conference on Pervasive Computing. London, UK: Springer-Verlag, 2002: 167-180.

[28] Wang W, Li W, Wu Z, et al. An Ontology-Based Context Model for Building Context-Aware Services[C]//Proceedings of the 2nd International Conference on Intelligent Systems, Modeling and Simulation. Washington, USA: IEEE Xplore, 2011: 296-299.

[29] Gruber T R. A Translation Approach to Portable Ontology Specifications[J]. Knowledge Acquisition, 1993, 5(2): 199-220.

[30] Hao Qian. Research on Context Modeling and Reasoning in Ubiquitous Learning Environment[D]. Dalian: Dalian University of Technology, 2011.

[31] Li W, Xia Q. A Method of Concept Similarity Computation Based on Semantic Distance[J]. Procedia Engineering, 2011, 15: 3854-3859.

[32] Liu Honglan, Gao Qingshi, Yang Bingru. Proposition Relativity and Logic Calculation in Many-Valued Logic[J]. Journal of University of Science and Technology Beijing, 2007, 29(S2): 172-177.

[33] Chen H, Tolia S. Steps Towards Creating a Context-Aware Agent System[R]. HP Laboratories Palo Alto, 2001.

[34] Zhang Chucai, Liu Junke, Qu Shaojun. A Context Reasoning Method Based on Ontology and User-defined Rules[J]. Computer Technology and Development, 2014, 24(4): 139-142.

[35] McGuinness D L, Harmelened F. OWL Web Ontology Language Overview[EB/OL]. [2004-02-10]. https://www.w3.org/TR/2004/REC-owl-features-20040210/.

[36] Horrocks I, Patel-Schneider P, Boley H, et al. SWRL: A Semantic Web Rule Language Combining OWL and RuleML[EB/OL]. [2004-05-21]. https://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

Author Contributions Statement

Xia Lixin: Conceived research ideas, designed research plan;
Yang Jinqing: Developed system, collected, cleaned, and analyzed data, drafted

manuscript;
Cheng Xiufeng: Revised final version of paper.

Conflict of Interest Statement

All authors declare no conflict of interest.

Support Data

Support data is self-archived by the authors, E-mail: yjq@mails.ccnu.edu.cn.
[1] Yang Jinqing. Mobile_{{{Data}}_{{Collection}}}.xlsx. Data collected by MDCF system.

Received: December 9, 2016
Revised: March 13, 2017

Data Analysis and Knowledge Discovery
ChinaXiv Partner Journal
Issue 5, 2017

Collecting Mobile Data Based on Content Awareness ——An Empirical Study

Xia Lixin, Yang Jinqing, Cheng Xiufeng
(School of Information Management, Central China Normal University, Wuhan 430079, China)

Abstract: [Objective] This paper proposes the framework for a mobile data retrieval and analysis system based on context-awareness, aiming to optimize the related data mining procedures. [Context] Nowadays, the automatic dynamic and comprehensive applications for mobile data mining were still being developed. [Methods] First, we proposed a framework to collect mobile data from the client side with the help of Android AWARE sensor. The collected data was received by the server automatically. Then, we designed an empirical study to analyze the retrieved APP usage data. [Results] The proposed system could effectively recommend useful APPs to the mobile users. [Limitations] More in-depth analysis was needed to examine the collected data. [Conclusions] The proposed framework could help us effectively retrieve and analyze mobile usage data, which benefits the contextual computing research and the mobile information behavior studies.

Keywords: Context Awareness; Application Framework; Android; Context Reasoning; User Behavior

Note: Figure translations are in progress. See original paper for figures.