Nunchi: Predicting Availability from Mobile Phone Usage

Taegyeung, Lee, YongJae Chang, DongSu, Lee
Korea Advanced Institute of Science and Technology
Daehak-ro 291, Yuseong-gu
Daejeon, South Korea
{chemidong, danny003, youngjae.chang}@kaist.ac.kr

Abstract— **We presents Nunchi, a system that enables to state human's availability to their smartphone. In order to make it, NunchitBab, an application that records smartphones' featured sensor data is used for data aggregation. We collected 2,000,000 logs of sensor data from three candidates during two weeks. Then, we apply a decision tree and a neural network model to the collected data. The result show 18% of precision and 54% of recall. Due to NunchitBab's low recall rate, Nunchi uses positive and negative pattern analysis to get user's availability. The pattern analysis of Nunchi uses an intentional notification and sensor data after notification arrived for 7 seconds. Based on Nunchi, Hillgt application is developed under Android and Firebase database server. Hillgt enables to know other user's availability into three levels (Available, Might Available, Not Available) with Nunchi's decision.**

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION (*HEADING 1*)

We live in the constant stream of information. Though we have extraordinary ability to sense, remember and reason, there is limit in our capability; we cannot process all the information. Particularly, psychologists have found that people selectively concentrate on a discrete set of information. Thus we assign scarce attentional resources to limited area ignoring other perceivable information. Such allocation become principal on the ability to memorize, multitask, or recognize targets in a noisy environment.

Nonetheless, these results indicate that we can only interact with a limited world where we are concentrating on. As such, we often generate implicit or explicit clues telling what we are selectively attending to and how deeply we are focusing. Researchers studying communication have found that accurately signaling and detecting attention state is a central key for fluent and fast interaction among people. Attentional cues help us decide when to speak and contribute to the conversation, and to detect the progress of conversation. Furthermore, by inferring other actors' attention we achieve *grounding* — having a common understanding behind communication.

The findings about the use of limited attention especially in communication shows detection of these attentional signals is essential for fluid collaboration. This also implies that design of computational agent or interface should consider attentional signals as a major input. The introduction of mobile smartphone have conquered big portion of everyday communication. However, many messaging apps are failing to accurately convey users' attentional signals. As the sensing ability of the smartphone is increasing we've tried to capture user's attentional cue and convey it via messaging system; eventually, building an attentional user interface.

To explore the opportunity, we focused on the conversation initiation pattern. The modern smartphone usage pipeline often starts with notification. When a service need an attention, i.e. Facebook mention or KakaoTalk message, the app generates notification putting it on the lock screen, notification bar and anywhere else. The notification usually accompanied with a buzz or screen wakening, generating a stimulus that can be perceived. We analyzed users response upon the notification and the result whether user attend to it or not. Based on the observation we extract an typical pattern that can be detected as an attentional cue. Finally, we built an application that can examine user's attentional states in distant environment.

II. RELATED WORKS

Most of research is done on managing notification on the smartphone. There was a recent workshop held in Mobile HCI '15, researching this area. [4] There are also works focused on implementing application which predicts user's attentiveness with context aware features. They decide whether a user will view a message in few minutes or not [1], or determine the best time to interrupt the user by predicting attentiveness [3]. The researches are based on the analysis of users' pattern and habits, heavily studied in Oulasvitra's paper [2].

III. METHODOLOGY

To build the Nunchi as a fine grained system, we aggregated data about selected features and designed a learning model that uses aggregated data as input to gain the Nunchi value. The data aggregation was designed as mobile application, NunchitBab, and it was deployed in Android phone to construct database. Candidates of machine learning model are evaluated with data from NunchitBab after aggregation. In this session, detailed explanation of

NunchitBab, selected features, and learning models will be descripted.

A. NunchitBab: Data Aggregation

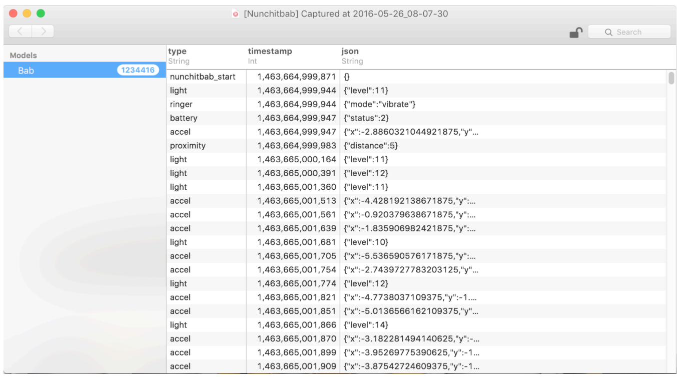
NunchitBab is an Android application developed in Android API level 19. It was deployed in Android mobile of three volunteers, and aggregated traces for 13 days. The NunchitBab includes an activity to manage data (initiation, deletion, and report via email), and a service that runs all day to track user's behavior according to selected features. The features used in NunchitBab is shown in Figure 1.

No	Features	Levels	Details
01	nunchitbab_start	-	Start of NunchitBab
02	Nunchitbab_end	-	End of NunchitBab (turning off phone)
03	light	Brightness (lx)	Ambient brightness value
04	ringer	Status	Ringer mode (silent / vibrate / bell)
05	screen	Status	Screen turn on / off
06	noti_posted	Notification Info	Posted notification info: id, package, post time, title
07	noti_removed	Notification Info	Removed notification info: id, package, post time, title
08	unlock	-	Unlock phone
09	proximity	Distance (cm)	Proximity from phone
10	battery	Status	Battery state (unknown, charging, discharging, not charging, full)
11	accel	Acceleration(m/s^2)	Acceleration of phone: x-axis, y-axis, z-axis, magnitude

Fig. 1. Table of features used in data aggregation

Since NunchitBab requires notification information for data aggregation, NotificationListenerService was used to receive broadcast of post and remove of notifications. Therefore, the main service of NunchitBab extended NotificationListener-Service which is provided in Android API, and broadcast receivers for ringer mode, screen on/off, unlock, and battery state are registered in the service. Also, sensors for proximity, light, and acceleration are registered in the service, and the sampling period is set as 1ms.

NunchitBab recorded data as Realm, a popular mobile database, with timestamp, type, and json columns. Timestamp indicates recorded time in milliseconds, type represents feature type, and json represents values in json format. Figure 2 is example of aggregated data from NunchitBab shown in Realm Browser.



type String	timestamp Int	json String
nunchitbab_start	1,463,664,999,871	{}
light	1,463,664,999,944	{"level":11}
ringer	1,463,664,999,944	{"mode":"vibrate"}
battery	1,463,664,999,947	{"status":2}
accel	1,463,664,999,947	{"x":-2.8860321044921875,"y"...
proximity	1,463,664,999,983	{"distance":5}
light	1,463,665,000,164	{"level":11}
light	1,463,665,000,391	{"level":12}
light	1,463,665,001,360	{"level":11}
accel	1,463,665,001,513	{"x":-4.428192138671875,"y":...
accel	1,463,665,001,561	{"x":-0.920379638671875,"y":...
accel	1,463,665,001,639	{"x":-1.835906982421875,"y":...
light	1,463,665,001,681	{"level":10}
accel	1,463,665,001,705	{"x":-5.536590576171875,"y":...
accel	1,463,665,001,754	{"x":-2.7439727783203125,"y"...
light	1,463,665,001,774	{"level":12}
accel	1,463,665,001,821	{"x":-4.7738037109375,"y":-1....
accel	1,463,665,001,851	{"x":-5.0136566162109375,"y"...
light	1,463,665,001,866	{"level":14}
accel	1,463,665,001,870	{"x":-3.182281494140625,"y":-...
accel	1,463,665,001,899	{"x":-3.95269775390625,"y":-1...
accel	1,463,665,001,909	{"x":-3.87542724609375,"y":-1...

Fig. 2. Example of Realm database constructed from NunchitBab

Features that is triggered by Android broadcasting (notification, screen, unlock, ringer, NunchitBab start/end) are recorded only when the service of NunchitBab is received broadcasting, and features from sensors (light, acceleration, proximity) are written in database only if the values change. Therefore, aggregated data (raw data) are resampled with constant sampling duration, 10(s). For each sample, feature values are same with last changed value, and some features are added: time-related features such as time since last screen on and number-related features such as number of screen on. Modified features of resampled data is shown in Figure 3.

No	Features	Levels	Details
01	accel	Time (ms)	Time since last acceleration is posted
02	battery	Status	Battery state (unknown, charging, discharging, not charging, full)
03	light	Brightness (lx)	Ambient brightness value in lux
04	lock	Time (ms)	Time since last lock (screen off)
05	noti_pending	#	Number of pending notification
06	noti_posted	Time (ms)	Time since last notification posted
07	noti_removed	Time (ms)	Time since last notification removed
08	ringer	Status	Ringer mode (silent / vibrate / bell)
09	screen_on	#	Number of screen on since last lock
10	unlock_duration	Time (ms)	Duration of last unlock ~ screen off

Fig. 3. Table of modified features after resampling Attention Model

Nunchi system adapted in learning model to predict user's attention based on aggregated data from NunchitBab. We evaluated two candidate models: decision tree model and neural network model. For these models, we assumed that smartphone users are in available state, might-available state, or not-available state, and available state indicates that a user's phone is in interval between between unlock and screen off. Evaluated learning models are designed to predict whether users are in not-available state or might-available state when they are not in available state. For ground truth, interval from settled time before unlock until unlock is occurred was set as might-available state, and others were set as not-available state. State of each samples were labeled according to ground truth rule in sampling step. Settled time which is maximum duration of a might-available state was set as 3 minute and 5 minute. In case of light feature, variation of value increased as the value got bigger; therefore, we wrapped value of light feature with logarithm function.

1) Decision Tree

The decision trees are designed in two ways: with scikit-learn library and without other classifying libraries in python. The reason why we implemented additional classifier without scikit-learn is to make code easy to modify depth limit of decision tree and value threshold of each nodes. The candidates for threshold of each features are generated by diving interval of each feature values with 10, and thresholds that provide the smallest entropy of generated child nodes are selected for each features.

2) Neural network

Neural network is also implemented as learning model for Nunchi. It is flexible for learning rate and hidden layer size, and evaluated various learning rate and hidden layer to get the best fit model. For better performance of neural network model, feature values are modified. Feature inputs whose level is status such as ringer mode and battery state are vectorized, and output states are also vectorized since state numbers (in case of ringer mode - 0: silent, 1: vibrate, 2: bell) do not have numerical meanings.

IV. RESULTS

By NunchitBab application, data are aggregated from three volunteers for total 13 days. The evaluated results of decision tree model are shown in Figure 4 and 5. Figure 4 shows cross validation error, precision, and recall of model to compare tradeoff between overfitting and performance of decision tree to decide the best depth limit. Figure 5 shows evaluation result of decision tree that used scikit-learn library with 3 volunteer's data.

Depth limit	3 min			5 min		
	Recall (%)	Precision (%)	Cross Validation Error (%)	Recall (%)	Precision (%)	Cross Validation Error (%)
10	0.0	-	12.0	0.0	-	17.7
20	4.0	22.6	13.2	7.5	39.0	19.2
30	4.2	6.0	19.5	11.9	20.4	26.2
40	16.2	9.4	23.0	19.7	23.1	28.0
50	27.8	12.1	26.1	31.6	22.1	30.4
60	40.6	15.2	27.4	37.1	29.2	31.5

Fig. 4. Recall, precision, cross validation error of decision tree model

	3 min			5 min		
volunteer	1	2	3	1	2	3
Recall (%)	45.7	39.8	24.2	55.0	46.5	39.4
Precision (%)	11.9	4.5	35.5	17.0	6.5	48.8

Fig. 5. Evaluation result of decision tree for three volunteer's data

Neural network model evaluation results are shown in Figure 6. We note that evaluation of neural network model did not work with third volunteer's data. The hidden layers of the model were set as two layers with size 100, and the learning rate was set as 0.7.

	3 min			5 min		
volunteer	1	2	3	1	2	3
Recall (%)	55.7	55.6	-	32.9	68.9	-
Precision (%)	9.7	4.9	-	18.0	6.2	-

Fig. 6. Evaluation result of neural network for three volunteer's data

V. HILLGT: A DISTANT GLANCING

Hillgt is a prototype application for predicting a user's attention based on Nunchi's availability decision. Hillgt's goal is to know an availability of other smartphone users in 10 seconds. In this section, we present Hillgt's architecture with a high level overview of Hilt architecture and a server architecture, and the application's designs with UI.

A. Architecture

Hillgt application mainly consist of two parts, an application and a server. Figure 1 shows an overview of Hillgt system. When a user requests an other's availability, the application sends this information to the database server. The server toss it to requested user's Hillgt application. The Nunchi in Hillgt application makes a notification to the user. Then, Nunchi collects sensor data responded from the notification for 7 seconds and states the user's availability based on the data. And the state is sent to requester's application through the server.

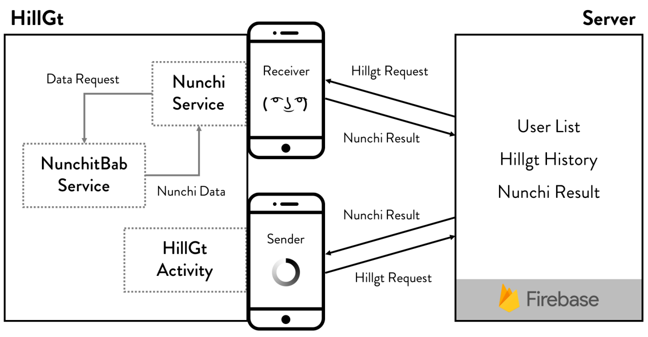


Fig. 7. High Level Overview of Hillght System

Application architecture consist of two parts, Nunchi and a server network.

1) Nunchi

Nunchi is a service that collects sensor data by making a notification and states a user's availability. Nunchi gather ringer mode, battery, screen, unlock, proximity, light, acceleration information using event listeners, broadcast receivers, and sensor listeners when values are changed. These sensor changes are divided into two actions which are a positive action and a negative action. If there is a distinct change of a sensor, for instance a difference of an acceleration is larger than 3.0, Nunchi consider the change is a positive action. But when ringer mode becomes a silent state, Nunchi consider the change is a negative action. If positive actions occurs, the state is available. If there is no change, the state is not available. And if a positive action occurs and negative action occur next, we define this state is might available.

2) Server Network

Firebase is used for a database server. The application send a request information to the server and gets the availability values. Figure 2 shows the architecture of the server. User information contain their ID and names. Every request are is recorded with a sender ID, a sender name, and a timestamp.

And a Nunchi value, which is user's availability, and total sent counts are also recorded with their ID and name.

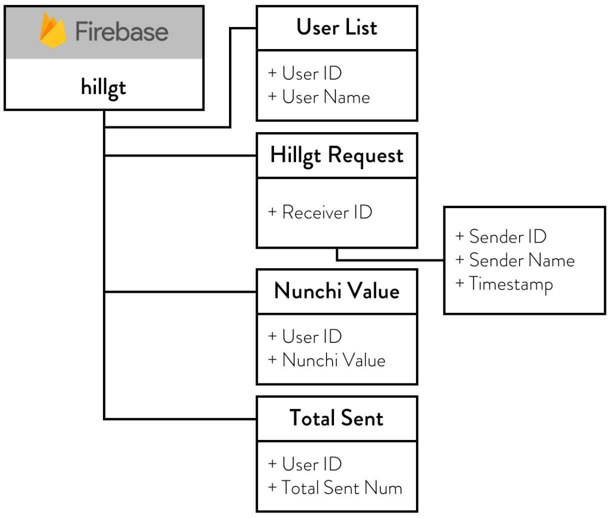


Fig. 8. Architecture of the Database Server

B. *Design*

This part shows real UI designs used in Hillgt. Hillgt's main pages shows list of friends. When a friend is clicked, it shows loading motion. After receiver sends the Nunchi value, this value is represented as a text right after the name. Figure 3 shows these sender's UI.

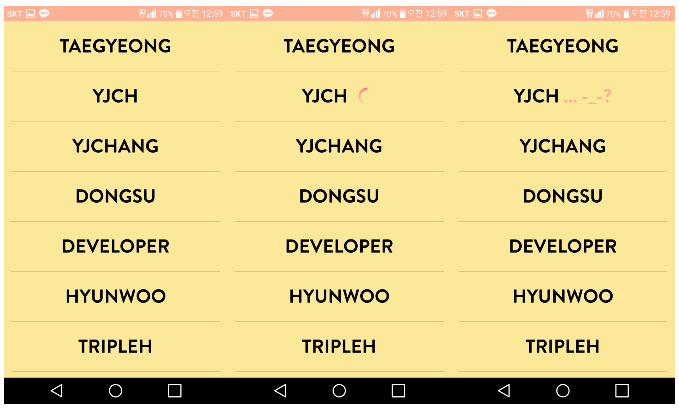


Fig. 9. UI of Sender's actions

When a receiver gets a request from others, a notification is appeared and makes 'Hillgt' sounds to alarm. And if moved to a right tap, total sent, received, received today counts are available to check. Also, logs of received request remains with a sender name and a sent time. Figure 4 shows a situation when a notification received and a right information activity.

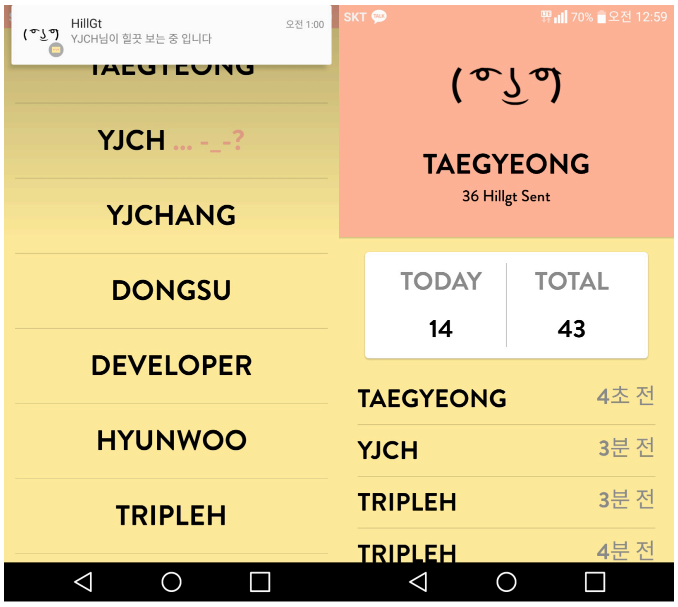


Fig. 10. Notification UI and Information activity UI

VI. Conclusion

In mobile communication situations, communicator's attention to their mobile device decides whether it is able to communicate or not. Unfortunately, smartphones don't know that their owner's attention. In this reason, people just predict or assume the other's availability. In order to solve this problem, we present Nunchi, a system that enables to state human's availability to their smartphone. We aggregate featured sensor data and did a decision tree and a neural network analysis. Since precision and recall shows lows results, Nunchi uses positive and negative pattern analysis to get user's availability. Based on Nunchi, Hillgt application is developed under Android and Firebase database server. Hillgt enables to know other user's availability into three levels (Available, Might Available, Not Available) with Nunchi's decision.

References

[1] G Pielot, Martin, et al. "Didn't you see my message?: predicting attentiveness to mobile instant messages." Proceedings of the 32nd annual ACM conference on Human factors in computing systems. ACM, 2014.

[2] Oulasvirta, Antti, et al. "Habits make smartphone use more pervasive." Personal and Ubiquitous Computing 16.1 (2012): 105-114.

[3] Pejovic, Veljko, and Mirco Musolesi. "Interruptme: designing intelligent prompting mechanisms for pervasive applications." Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing. ACM, 2014.

[4] Poppinga, Benjamin, et al. "Smarttention, Please! Intelligent Attention Management on Mobile Devices." Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct. ACM, 2015.

[5] Horvitz, Eric, et al. "Models of attention in computing and communication: from principles to applications." Communications of the ACM 46.3 (2003): 52-59.

[6] Pielot, Martin, et al. "When attention is not scarce-detecting boredom from mobile phone usage." Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing. ACM, 2015.

[7] Dingler, Tilman, and Martin Pielot. "I'll be there for you: Quantifying Attentiveness towards Mobile Messaging." Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services. ACM, 2015

[8] Chang, Yung-Ju, and John C. Tang. "Investigating mobile users' ringer mode usage and attentiveness and responsiveness to communication." Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services. ACM, 2015.

[9] Okoshi, Tadashi, Jin Nakazawa, and Hideyuki Tokuda. "Attelia: sensing user's attention status on smart phones." Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication. ACM, 2014.