People's Interruptibility in-the-wild: Analysis of Breakpoint Detection Model in a Large-Scale Study

Kota Tsubouchi
Yahoo Japan Corporation
1-3 Kioicho, Chiyoda-ku,
Tokyo 102-8282, JAPAN
ktsubouc@yahoo-corp.jp

Tadashi Okoshi
Graduate School of Media and Governance, Keio University
5322 Endo, Fujisawa
Kanagawa, 252-0882, JAPAN
slash@ht.sfc.keio.ac.jp

UbiComp/ISWC'17 Adjunct , September 11–15, 2017, Maui, HI, USA
ACM 978-1-4503-5190-4/17/09.
https://doi.org/10.1145/3123024.3124556

Abstract
In the advancing ubiquitous computing where users have been having an increasing amount of push-style information provision from lots of intelligent proactive services, detecting the users' current attentional status and/or interruptibility has been a significant issue. In our previous study[7] on interruptibility detection in "Yahoo! JAPAN" real world product for 21 days with more than 680,000 users, the result showed significant lower user response time in exchange for notification delivery delay for about 4 minutes on average. In this paper, we further analyze the features in the interruptibility detection model trained and updated during this study in nightly basis for 21 days and report the results.

Author Keywords
ubiquitous computing; attention; interruptibility; research community; opportunities

ACM Classification Keywords
H.3.4 [User profiles and alert services]: Systems and Software

Introduction
The amount of information available for consumption has been growing by several orders of magnitude, while the capacity of our attention as humans is constant. For better

timeliness and speediness, provision of information from versatile types of applications and services have been becoming more proactive. Delivery of such proactive information is is often done through push notification systems. In this information-overload world, the constant and limited capacity of human attention has become a new bottleneck [2] in computing. Various past literatures already revealed the negative effects caused by divided attention, usually caused by the push notifications, in terms of productivity, emotion, and mental state [3, 11, 10, 1].

In recent ubiquitous computing research, researchers have been investigating users' interruptibility with different techniques and actual detection targets, such as breakpoint [4], interruptibility, and boredom [8, 5, 9].

Based on our past research results on real-time breakpoint on mobile and wearable devices in ubiquitous computing situations [5, 6], our next significant research motivation was to investigate if such methodology actually works effectively in the real world environment. We design and implemented the same interruptibility detection and notification scheduling in "Yahoo JAPAN" Android application [1], one of the most popular Android applications in the Japanese market with more than 10 million install base. Results from our large-scale user study with 687,840 users for 21 days revealed that, in exchange for average notification delivery delay of 4 minutes due until an immediate detected breakpoint, users' response time to the delivered notification is significantly reduced (49.7%). The analysis also revealed higher click rate and user engagement level throughout the entire study period.

In this paper, we particularly focus on the breakpoint detection model that was trained and periodically updated during the user study. Before the user study was initiated, an initial breakpoint detection model was trained in our initial small model-training experiment with 39 devices for 35 days. That initial model was installed into all the user's smartphone at Day 1 of the user study. Once the user study was began, logs from all the clients were sent to our server and a new model was trained and delivered to all the clients in nightly basis. Thus, throughout the 21 days of the study, every day users used a new model that was just trained last night. In this paper, we investigate deeply inside such model of the study and evaluate them whether the created model are updated intuitively or not.

System Architecture

Figure 1 shows the architecture of our production system [7]. Our implementation consists of a series of additional components inside the Yahoo! JAPAN Android application as well as the components on the server side.

The **Mobile Sensing** component obtains several types of sensor data, including that from the GooglePlay Service Location API `ActivityRecognition` and other device-related data. The data mainly consists of a series of output values from the GooglePlay Service Location API's "ActivityRecognition" and other device-based events (e.g., screen on/off events). On the activity recognition results from "ActivityRecognition", we used data only with a confidence value greater than 51 (the value can be between 0 and 100) based on our empirical knowledge. The mobile sensing component obtains all of these data through individually implemented event handlers for each sensor. Any changes in sensor data, such as when the user's activity changes or when the device volume is changed, will be sensed and logged by the system.

[1] https://promo-mobile.yahoo.co.jp/yjapp/

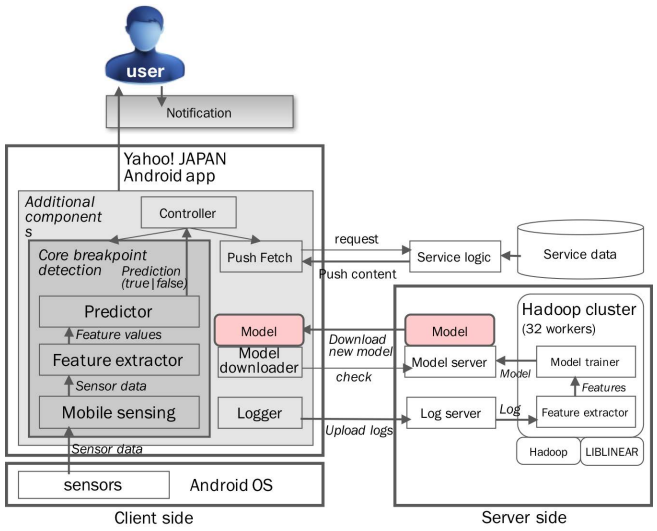

Figure 1: System Architecture

When new sensor data is detected, the **Feature Extractor** and **Predictor** modules execute and predict if the current moment is a breakpoint of the user. The total number of features is 387. The system extracts seven types of features: timestamp (hour), activity type, volume, device sleep/awake status, vibration status, silent mode setting, and network connection type. In addition to these sensor values, transition of the sensor data is introduced as an eighth type of feature. For all possible "From" and "To" pairs of sensor status transition, we prepared a dedicated feature value. To detect breakpoints that occur in the timings of activity change (e.g., "tilt"ing their phone from the "still" status), these transition type features are helpful for the system to characterize changes detected in each sensor.

Using the above features, the actual breakpoint prediction is executed in **Predictor** with an installed linear regression model. The model parameters are generated on the server side and then downloaded to the clients.

The log data along with the ground truth annotation will be sent to the server nightly. At the server side, every night a new model is built from all the data uploaded from the clients in the past. The resulting model's parameter will be available on the **Model Server** and will be downloaded to the clients on a daily basis. This scheme of the periodic model update and distribution nicely fits our system design requirement that modification in the production client software need to be minimized.

Experiment Overview

On the basis of promising results from our initial study [7], we conducted a large-scale in-the-wild user study in the production environment with 687,840 users for three weeks to better understand how our breakpoint-based notification scheduling works in a real user environment.

The user study was conducted for three weeks (21 days) in September 2016. To ensure the stability of the production application, the new version (including our implementation) was released to the production environment with a graduated deployment scheme on the app store. After three days, the new version was made available for all users.

As written in our previous paper [7], we found through experiment that, in most cases, notification delivery delay due to breakpoint detection does not hurt and even improves a user's overall click timing (earlier), with significantly reduced user response time (49.7%). We also observed a continuous increase in content click numbers and user engagement level over the entire study period.

Further Analysis of Model

In this paper, we further investigated the model updated daily in the above experiments in more detail. The model is an array composed of the weight in each of the 387 features extracted from the sensor data and device configuration. The breakpoint detection is judged by the linear sum of the weights of the features at the present time obtained from the user's smartphone. Therefore, the larger the value of the weight means, the more the features of the object is effective information for judging breakpoint.

Transition of Weight Values

Fig.2 shows the weight score of each feature during the 21 days of the experiment. The X axis shows the number of days elapsed since the start of the experiment, while the Y axis the weight score of each feature. The days with orange background color represent holidays, while the others indicate week days. At Day 1 (X=0), the initial model the users used was a preliminary model previously trained from the data collected from 39 subjects beforehand.

The following two considerations are described in the result of the transition of the weight score shown above. The first point is that the weight scores of almost all features are saturated in about beginning 3 days. While some scores keep increasing or decreasing gradually, many of them tend to settle around 3 days after the start of the experiment. Even though it goes up and down depending on the day, the basic relationships and tendency among different features tend to be preserved. As the number of users is as large as about 340,000 people (for the experimental group), it turns out that the model has stabilized in a small experiment period.

The second point is differences between weekdays and holidays. The absolute value of the weight score of each feature tends to be large on holidays. Also, the ranking (order)

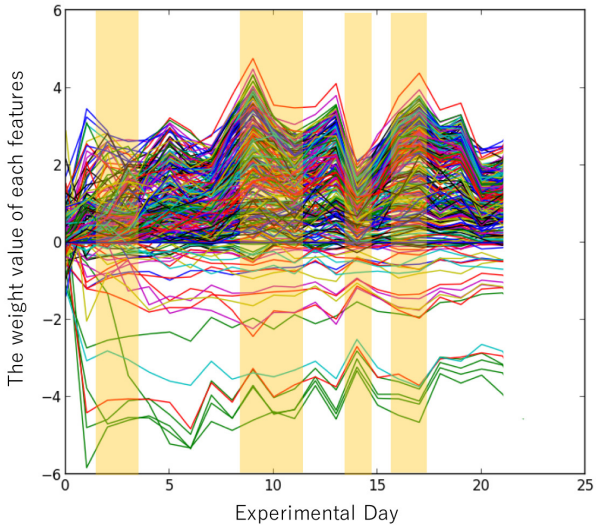

Figure 2: Transition of the weight values

of features tend do change in many holiday cases. We consider that these facts indicate (1) the model gets unstable and (2) some additional different training data have been added to the model. From this analysis, we conclude that, in the future experiment, it is better to separate the model into two different models, namely one for weekdays and another for holidays.

Weight Value Changes Over Study Period

Table 1 shows the difference in the weight value between the first day and the last day of the experiment. The "volume" means the volume that the user selected for the ringtone of the smartphone. The upper table shows the top 10 features whose weight value increased from the first day to the last day, and the lower table shows the features of top

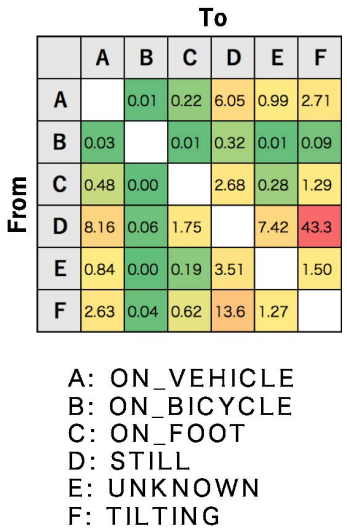

Figure 3: Breakdown of "True"-labeled Detected Breakpoints into Activity Changes.

10 whose weight value decreased.

data	from	to	dif. of score
volume	3	4	4.222
volume	1	0 (silent)	3.745
volume	1	2	3.477
volume	6	7 (max)	3.211
activity	STILL	TILTING	3.195
activity	TILTING	UNKNOWN	3.186
activity	UNKNOWN	STILL	3.178
activity	UNKNOWN	ON_FOOT	3.170
volume	2	1	3.158
volume	4	5	3.119

data	from	to	dif. of score
activity	IN_VEHICLE	IN_VEHICLE	-4.713
activity	ON_BICYCLE	ON_BICYCLE	-4.560
trigger	none	volume	-4.407
activity	UNKNOWN	UNKNOWN	-4.350
activity	TILTING	TILTING	-3.583
activity	ON_FOOT	ON_FOOT	-3.391
trigger	none	others	-2.603
trigger	none	activity	-2.424
network	WIFI	WIFI	-1.713
network	LTE	LTE	-1.624

Table 1: The Difference in the Weight Value between the First Day and the Last Day. We can confirm the feature transition from one value to itself, e.g. "from LTE to LTE". It shows that the other factor such as activity mode was change, then the feature value of "from LTE to LTE" become "1".

On the upper table, we can see that increment of the weight score for the timing at which the user changes the volume of the smartphone and the timing at which the activity changes is large. It is an intuitive result that the increment of the weight score is larger when increasing the sound, such as volume 3 to volume 4, rather than decreasing the sound as to change the volume from 3 to 2, for example. When the user reduces the sound, we consider that the user's attention is often leaving from the smartphone, for example by starting the office work or other intensive activities. Also, Fig3 shows a breakdown list of detected breakpoints with "true" annotation (i.e., breakpoints with a notification that was clicked by the user within 10 seconds) into activity change pairs. Very interestingly, the features related to the activity in which the increment of the feature weight score was large, which is the state changed from STILL to TILTING, is a features which was also confirmed in "True"-labeled Detected Breakpoints.

On the contrary, the bottom table shows a list of features whose weight score has been greatly reduced. Everything listed in the table are features that does not show behavior or state transition. In other words, the models have been updated gradually no to classify cases without any activity changes as breakpoints. This result is also very intuitive.

Conclusion

We investigated deeply inside calculated model of breakpoint detection and evaluated the model whether it is updated intuitively or not. As a result of the verification, the results of the following three points were clarified.

- Thanks to the huge number of log data obtained from the enormous number of users, it only takes about several days for the model to stabilize.
- Models learned from log data containing both weekdays and holidays has become unstable. Therefore, the model should be generated and used respectively for weekdays and holidays.

- From the investigation on features whose score increased and decreased over the study period, we can conclude that the actual generated model was with quite intuitive feature construction.

REFERENCES

1. Mary Czerwinski, Edward Cutrell, and Eric Horvitz. 2000. Instant messaging: Effects of relevance and timing. In *People and computers XIV: Proceedings of HCI*, Vol. 2. British Computer Society, 71–76.
2. D. Garlan, D.P. Siewiorek, A. Smailagic, and P. Steenkiste. 2002. Project Aura: toward distraction-free pervasive computing. *Pervasive Computing, IEEE* 1, 2 (april-june 2002), 22 –31. DOI: http://dx.doi.org/10.1109/MPRV.2002.1012334
3. J. G. Kreifeldt and M. E. McCarthy. 1981. Interruption as a test of the user-computer interface. In *JPL Proceeding of the 17 th Annual Conference on Manual Control*. 655–667.
4. Darren Newtson and Gretchen Engquist. 1976. The perceptual organization of ongoing behavior. *Journal of Experimental Social Psychology* 12, 5 (1976), 436–450.
5. Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K. Dey, and Hideyuki Tokuda. 2015a. Attelia: Reducing User's Cognitive Load due to Interruptive Notifications on Smart Phones. In *Proceedings of IEEE International Conference on Pervasive Computing and Communications 2015 (PerCom '15)*.
6. Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K. Dey, and Hideyuki Tokuda. 2015b. Reducing Users' Perceived Mental Effort Due to Interruptive Notifications in Multi-device Mobile Environments. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. 475–486.
7. Tadashi Okoshi, Kota Tsubouchi, Masaya Taji, Takanori Ichikawa, and Hideyuki Tokuda. 2017. Attention and engagement-awareness in the wild: A large-scale study with adaptive notifications. In *Proceedings of I2017 IEEE International Conference on Pervasive Computing and Communications (PerCom 2017)*. 100–110. DOI: http://dx.doi.org/10.1109/PERCOM.2017.7917856
8. Veljko Pejovic and Mirco Musolesi. 2014. InterruptMe : Designing Intelligent Prompting Mechanisms for Pervasive Applications. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '14)*. ACM, New York, NY, USA, 395–906. DOI: http://dx.doi.org/10.1145/2493432.2493445
9. Martin Pielot, Tilman Dingler, Jose San Pedro, and Nuria Oliver. 2015. When Attention is Not Scarce - Detecting Boredom from Mobile Phone Usage. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15)*. 825–836.
10. Cheri Speier, Joseph S Valacich, and Iris Vessey. 1999. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences* 30, 2 (1999), 337–360.
11. Fred RH Zijlstra, Robert A Roe, Anna B Leonora, and Irene Krediet. 1999. Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology* 72, 2 (1999), 163–185.