International Journal of Advanced Research in Science Management and Technology
IJARSMT

"An Intelligent Notification System Using Context from Real-Time Self Activity Monitoring System"

Dhananjay Behre[1], Prof.G.M.Poddar[2], Prof.M.R.Tiwari[3]

UG Scholar, Dept. of Computer Engineering, GCoE Nagaon, Dhule,M.S., India[1]

HOD, Dept. of Computer Engineering, GCoE Nagaon, Dhule, of, M.S.,India[2]

Asst Prof., Dept. of Computer Engineering, GCoE Nagaon, Dhule, of, M.S.,India[2]

Abstract- People can now receive custom-made information through Smartphones, tablets or wearable devices. However, people often tend to miss vital information, even reminders, in the flood of notifications. The problem of finding convenient moments for need-to-know information should be investigated. Because each person's message awareness pattern on a smart medium might be different, the necessity of personalized notification time should be emphasized. We believe that tracking changes in a user's physical activity and other contextual factors will reveal the most convenient moments. We propose a mobile framework, smartNoti, to carefully examine the user environment. The main contributions of our framework are: 1) developing an architecture to provide crucial information in a timely manner at a recognizable moment; 2) integrating, processing, training, and storing personalized latent features from heterogeneous data streams; 3) detecting user context transitions that might provide recognizable and available moments; and 4) predicting these moment and providing a notification message. The experimental validation on Intelligent Callback Reminder, which we implemented onan android application to notify a user missed or rejected call, demonstrates that our approach is effective. W e believe that our findings can lead to intelligent strategies to issue unobtrusive notifications on today's smart phones at noextra cost, by using sensors and contextual factors.

Keywords- Notification system, context awareness, contexture modeling, user logging, moment prediction.

I. INTRODUCTION:

The Intellectualization of the notification systems is a principal challenge in the Datafication [1] world. The emergence of so-called cyber-physical and cyber-physical- social systems [2] augments the necessity of suitable notification times. For example, the online mass media pushes breaking news to application software users. Social network systems notify users whenever updated events occur. Calendar events send alarms at the planned time. Even fitness programs press users to achieve their daily goals. On one hand, these notification messages can be helpful, on the other hand, people might soon be overwhelmed with the huge amount of notifications. To avoid the deluge of information at the involuntary times, notification systems need to be made aware of their current environment. Infrastructures for understanding user environments, such as wearable sensors, smart phones, and network systems have grown in use. Therefore, now is an ideal moment to explore and design the timeliness in notification systems.

The challenges of context-base notification systems are: selecting fundamental context factors that influence the activity patterns of individual agents; keeping track of these factors in real-time; carefully looking into the monitoring data; and immediately coping with recognized desirable situations. Considering these challenges, we propose a mobile framework to provide a standardized method for finding the recognizable and available moment. "R e cognizable moment" signifies a time to be noticed by the user. "Available moment" means a time that the user can carry out the suggestions made by notification message. We propose the hypothesis that some appropriate moments will exist at context transition moments. We define context in the same way as that A.K Dey et al. defined it in [3], apply the cybernetic principles [2] into the mobile framework, and then suggest a method to predict the moment. To demonstrate our approach, we implement a mobile

notification application called Intelligent Callback Reminder on smartNoti. The applicability of this approach is demonstrated on a case study of the smartNoti system focusing on 6 users over a period of about two months. The application details are as follow: Purpose • To notify the user of callback reminder messages in the case of missed/rejected calls at the recognizable and available moment. Scenario • smartNoti keeps track of user contexts. When a missed/rejected call occurs, smartNoti starts detecting context transitions based on real-time user logging. For example, a user receives a missed call during a group meeting. After the meeting, the context transition between meeting and chatting is caught by smartNoti. The system judges whether the moment is recognizable and available for the notification. If it is an ideal moment , smartNoti notifies the user with callback reminder message. If it is not an appropriate time, smartNoti keeps detection process running.

The main goals of our proposed framework are: to provide an architecture to notify crucial information in a timely manner at a recognizable moment; (i) integrate, process, train, and store personalized latent features from heterogeneous data streams; (ii) detect context transitions that might constitute recognizable and available moments; (iii) predict these moment in a timely fashion; (iv) and provide notification messages. This line of research is important because there are large amounts of messages trying to get in touch with users. To find the best notification moments, we assume that individuals have their own preferred moments to receive notifications and these will be one of context transition moments.

II. LITURATURE RIVIEW

Research on notification systems that investigates appropriate moments has become popular. However, to the best of our knowledge, there is no specialized mobile framework to find both recognizable and available moments through a total directional approach – real time chronological observation and analysis of user contexts, and a statistical model. Current studies try to minimize user interruption by choosing the best moments. While there are many factors that influence a person's interruptibility, the literature has found current user activities to be the most relevant. Interruptions between coarse breakpoints, i.e., major changes in workflow, produce less annoyance, and users have assessed them as being more respectful of their ongoing activity than random moments [7, 8]. To detect and differentiate between these breakpoints in interactive tasks, models can be created. Ho and Intille [9] used two accelerometers and activity recognition to trigger interruptions at moments when users change the activity, e.g., when from sitting to walking. They showed that users are significantly more receptive to interruptions than a random condition. Kern and Schiele used wearable accelerometers, audio, and location sensors to decide whether a user should be notified and which modality to use. They argue that no advanced model is needed, as the combination of tendencies is already sufficient [10].

III. METHODOLOGY

Figure 1 shows the architecture of smartNoti. By following the cybernetic principles [2], building models, measuring parameters, estimating states, and controlling the system, we divide our architecture into three layers: logging, processing, and prediction. **Logging layer -** Measuring parameter. The logging layer contributes to two main tasks: context factor logging and event factor logging. Context factor logging integrates six different categories: activity (physical activities and number of steps), location (latitude, longitude, and venue), time (week, weekend, or dawn, morning, afternoon, evening), logical (calendar), entertainment (audio, video, application usage), and system (silence, vibration, bell). More categories can be dynamically added as needed. Some examples of event factors are incoming/outgoing calls or missed/rejected calls. **Processing layer** - Building models. Context factors are processed every 5 minutes in this layer. Processed data is used to build a context matrix the time-window. Whenever event factors occur, the processing layer builds the user model using time-window at that point. Time-window is stored in Temporary Data for only 5 minutes. If event factors occur within this time, Model Learner uses the stored time-window and trains the model. The stored time-window is updated every 5 minutes.

The Naïve Bayesian Model is maintained in a local path as a type of SQLite database with necessary data such as average, standard deviation, and the sum of squares of differences from the (current) mean (m2). The model does not keep all of its index data, but rather uses the online algorithm of [5,6], and updates incrementally to calculate the posterior probability.

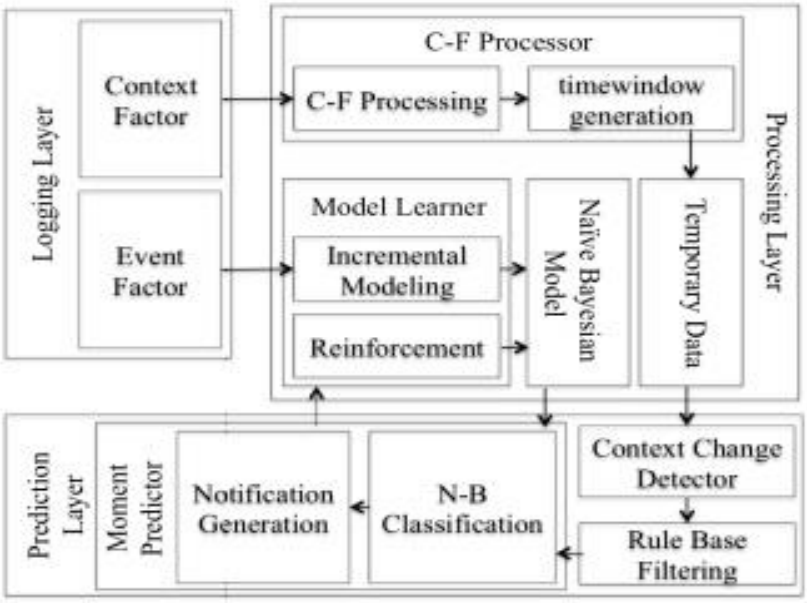

Fig.1 Architecture of smartNoti system.

Prediction layer 1 - Estimating states. Context-Change-Detector (C-C-D) estimates the current state by calculating the distance between target time-window and incoming time-window. "Target" means the time when event factors occur, and "incoming" means every 5 minutes. We estimate whether the incoming user context has changed compared to the target context. The estimation algorithm is described in Algorithm 1. Prediction layer 2 – Rule base filtering. smartNoti passes the incoming matrix through a pre-defined rule filter, and handles each context by following four different rules: release, skip, direct notification, and maintenance. Release means to stop tracking the moment, skip signifies to ignore the current context and wait for new incoming. Direct notification means to immediately notify, and maintenance means to keep tracking the moment. Each category can easily include additional rules. Prediction layer 3 - Controlling the system. If C-C-D detects context transition moment, the Moment Predictor predicts whether the detected time is the recognizable and available moment, and then generates notifications.

IV. RESULTS

Chronological observation and analysis of user context gives insight into which moments should be focused on. We are confident that there must be a suitable notification moment between the i^{th} context and the $(i \square 1)^{th}$ context. For example, when a person who has been working in an office moves to go to the restroom, we detect that transition and check the moment through a model. In particular, we try to figure out the most timely moment among all those of context transitions. The method for detecting context changeis to generate time-windows first and then to catch the transition by calculating the distance between the target window and the incoming window. The most important aspects of calculating distance are: what kind of context factors should be utilized; how to simply but effectively compare two contexts; how to deal with a different range of continuous values and harmonize with discrete values; and how to personalize the transition threshold. This is describedin the following sections and in Algorithm 1.

Algorithm 1 Context Change Detection Algorithm

Input: T_i □ □$F_1, F_2, F_3, ..., F_n$ □;

// 1 □ i □ 288 , n = the number of context factors

Output: TRUE, FALSE

1: **if** C-C-D is tracking context change

2: Calculate distance between T_k and T_i ;

// T_k = Target time-window,

// T_i = Incoming time-window3: **if** distance > threshold

4: **return** TRUE;

// Detect context change, and stop tracking5: **else if** maximum tracking time up

6: **return** TRUE;

// Stop tracking

7: **else** i++;

8: **return** FALSE;

// Wait next time-window, keep tracking

9: **endif**

10: **else if** T_i == tracking event factor

11: Set T_i to target time-window T_k , start tracking;
12: i + + ;
13: return FALSE;
14: **endif**;

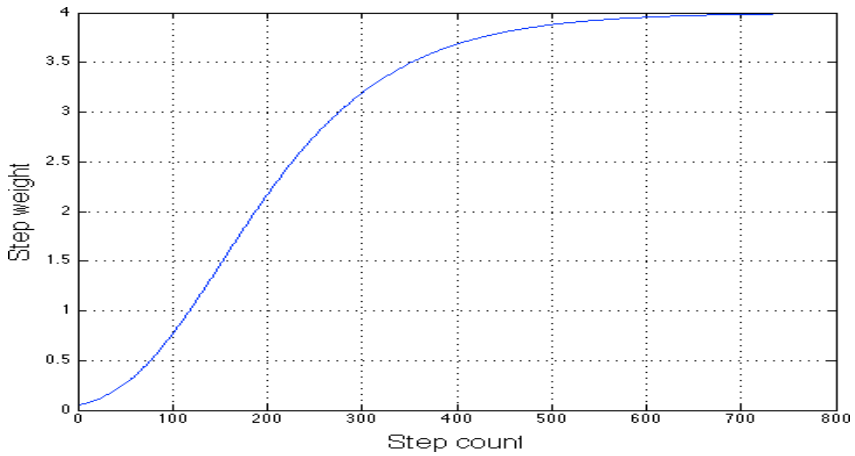

Fig.2 Generalized logistic curve of step counts.

K = 4, A = 0, Q = 0.01, B = 0.01, M = 150, v = 0.01

Latent features

We divide the latent features into continuous variables and discrete variables. Continuous variables include activity

level, step count, and amount of application usage. We define activity level as average intensity of accumulated physical activities. Each physical activity type, (standingstill, tilting/unknown, on foot, on bicycle, in vehicle) is scored from 0 to 4 by its intensity; its frequency during five minutes is counted and then the level is calculated in (1) where i is the intensity of the activity and k is the unique number of accumulated physical activities. Step count and the amount of application usage needs tobe normalized. Figures 2 and 3 show the generalized logisticcurve [4] of step count and the amount of unique applicationusage. By heuristically normalizing these factors with the generalized logistic function (2), we try to reasonablyconvert original values to normalized values.

Time-window

We define time-window as a $1 \square N$ matrix, which divides a day into 288 contexts. Processed context factors generate a time-window every 5 minutes, and it is stored temporarily until the next processed factor occurs. Each time-window is given a unique ID by its chronological order. Table 1 shows some time-window samples. Each row signifies a context, which is composed of n number of context factors. Time- windows are used to train the user model as well as to detectcontext change by comparing the distance between targetand incoming.

Context change detection

We detect a context transition by calculating L2-norm Euclidean distance, distance(T,C), between target time- window and incoming time-window. To start the C-C-Dprocess, first, we have to set a negative event on *smartNoti*, which becomes a trigger to start this process. For example, we set a missed call and a rejected call as negative events.

INCREMENTAL NAÏVE BAYES MODELING, CLASSIFICATION, AND NOTIFICATION

In this section, we design an incremental user model using the Naïve Bayesian (NB) approach as a baseline to identify whether the statistical model can be utilized in predicting user events. As NB only needs to update a few entries in its probability table, it consumes a much lower cost than a non-incremental approach, which has to rebuild a new classifier to include new training data. The real-time processing system running on a mobile device needs to consider not only computational cost, but also its limited memory space. We incrementally train the user model whenever events occur, and keep only a small amount of information in order to calculate posterior probability. The total count of discrete attributes is accumulated, and the online algorithm [5,6] helps to incrementally train continuous attributes by maintaining only three components; average, standard deviation, and m2. To classify an incoming context Ti as a suitable or unsuitable moment, we label each event with two hypotheses, H0 (positive) and H 1 (negative), and choose a maximum posteriori hypothesis by (5).

P(H0 | F1,...,Fn) > or < P(H1 | F1,...,Fn) (5) Posterior probability is shown in (6). P(H0 | F1,...,Fn) ≈ P(H0) P(Fi | H0) i=1 By assuming that continuous attributes follow Gaussian distribution, posterior probability is calculated by (7). The probability of discrete attributes is (8). P(Fi = fi | H0) = 1 σ 2π e − (Fi−μ) 2 2σ 2 (7) P(Fi = fi | H0) = the number of fi in H0 total number of H 0 (8) Lastly, the maximum posteriori hypothesis is calculated by log-likelihood ratios (9). If the context is closer to positive hypothesis H0 , as in the case of incoming or outgoing calls, we immediately pop up a notification message, but if it is more close to H1 , we skip the context, and wait for another one.

Log P(H0 | F1,...,Fn) P(H1 | F1,...,Fn) = log P(H0) P(H1) + log i=1 n ∑ P(Fi | H0) P(Fi | H1) (9)

EXPERIMENTAL VALIDATION: INTELLIGENT CALLBACK REMINDER

To evaluate smartNoti, we implemented the callback reminder application on an Android. Goal. T o notify the user of callback reminder messages in the case of missed/rejected calls at the best suitable moment. Measuring parameter. Context factors (physical activity, step count, the amount of unique application usage, media usage, time, week/weekend), event factors (incoming call, outgoing call, missed call, rejected call), target factors (missed call, rejected call). Building

model. Whenever event factors occur, the time- window at the incoming, outgoing call moment is labeled as positive class, and the time-window at the missed, rejected call moment is labeled as negative class. Estimating state. To detect a context transition moment, the system compares the time-window at the missed/rejected call moment to the incoming time-window. Controlling the system. The model predicts whether the transition moment is a recognizable and available time, and then sends the reminder message.

V. CONCLUSION:

Finding a suitable moment is an important problem of the notification system. To completely convey need-to-know information, a moment should be immediately recognizable and available. However, most research on notification systems concentrates on either the recognizable moments or the message themselves. To solve this limitation, we proposed smartNoti, a mobile framework, which integrates, processes, trains, and stores personalized latent features, and detects context transition that might be the recognizable moments and predicts the moment availability. As part of identifying the effectiveness of the statistical model in moment prediction, we proposed an incremental NB model and demonstrated the potential of using an incremental learning algorithm. To evaluate the moment reasonability and efficiency of the personalized data model, we implemented a callback reminder on the smartNoti framework. The application was evaluated using two methods, a model prediction test and a user feedback test. In our future work, we will investigate the influences of other context factors such as location factors, logical factors, and optimize the proposed user model by applying these additional features and comparing the Naïve Bayes algorithm to other incremental classification algorithms.

REFERENCES

[1] **J. A. Nelder, "The Fitting of a Generalization of the Lo-gistic Curve," Biometrics, vol. 17, no. 1, pp. 89–110, Mar. 1961.**

[2] **Tony F. Chan, Gene H. Golub, and Randall J. LeVeque, "Algorithms for Computing the Sample Variance: Ana-lysis and Recommendations," The American Statisticia-n, vol. 37, no. 3, pp. 242–247, Aug. 1983.**

[3] **Robert F. Ling, "Comparison of Several Algorithms for Computing Sample Means and Variances," Journal of the American Statistical Association, vol. 69, no. 348, pp. 859–866, Dec. 1974.**

[4] **Piotr D. Adamczyk and Brian P. Bailey, "If Not Now, when?: The Effects of Interruption at Different Mom e n-ts Within Task Execution," in Proceedings of the SIGC-HI Conference on Human Factors in Computing Syste-m s , New York, NY, USA, 2004, CHI '04**

[5] **Brian P. Bailey and Shamsi T. Iqbal, "Understanding C-hanges in Mental Workload During Execution of Goal directed Tasks and Its Ap.**