Inferring Presence Status on Smartphones: The Big Data Perspective

Aleksandar Antonić, Ivana Podnar Žarko, Domagoj Jakobović
University of Zagreb
Faculty of Electrical Engineering and Computing
Zagreb, Croatia
Email: {aleksandar.antonic, ivana.podnar_zarko, domagoj.jakobovic}@fer.hr

***Abstract*—In the context of communication services, presence is defined as the willingness and ability of a user to communicate across a set of devices with other users, and thus an up-to-date user presence status represents an essential prerequisite for real-time communications. Smartphones are a rich source of presence-related contex information, however; this information is currently not applied by the prevailing over-the-top communication systems to implicitly change user presence status in accordance with his/her context and typical daily behavior. Smartphone battery limitations and the abundance of context data generated from built-in sensors and mobile applications are the major factors limiting the adoption of rich presence solutions in state-of-the-art communication solutions.**

This paper presents an approach to learning and inferring user presence status on smartphones using the available context data with a goal to enable non intrusive and energy-efficient maintenance of presence status without user intervention. We apply the Mobile Data Challenge (MDC) data set collected during the Lausanne Data Collection Campaign from October 2009 until March 2011 in our evaluations.

***Keywords*-presence service; rich presence; supervised learning; Logistic Regression;**

I. Introduction

Presence service is often referred to as the dial tone of the 21st century since it enables users (i.e. watchers) to subscribe to presence information generated by their contacts (i.e. presentities), and to receive their presence updates in real-time [1]. Presence status is typically changed either explicitly based on direct user intervention or implicitly, for example, when a device running a presence agent is idle for a certain period of time. Over-the-top communication systems, e.g. Skype and GTalk, naturally integrate presence as one of its essential services, and are increasingly used on smartphones. However, existing smartphone implementations currently support only explicit change of presence status which requires direct user intervention. Moreover, if a Skype or GTalk client is running in the background on your smartphone while it is idle for a longer period of time, the presence status visible to your contacts remains ‘available’ regardless of your current context.

The paper proposes an approach to continuous learning and inferring user presence on smartphones which can automatically update user presence as well as free users from the annoying manual task of changing his/her presence status, such as putting it into the silent mode. Smartphones are a rich source of context data related to user presence. In particular, a call log, ring tone status and calendar events are good indicators of whether a user is available or unavailable for communication. For example, one may conclude that a user is available for voice communication when he/she accepts incoming calls while there are no scheduled events in the calendar. On the contrary, the user is either busy or unavailable when the phone is in the silent mode or when he/she rejects an incoming call. The patterns of presence changes are individual for each user. For example, some users ignore phone calls while driving or running, while others are at certain locations available only for texting. Next, some users accept phone calls during meetings but only from certain contacts, while others reply to missed calls via messaging. Thus the goal of our learning algorithm is **to identify personal presence-related patterns of user behavior** for each individual user, and **to infer user presence status such that it can be changed implicitly over time**. In addition, we would like to identify presence-related features which are energy friendly.

Presence management is an important service for real-time communication, and both the industry and standardization bodies have invested significant efforts over the last decade into the process of developing a suite of services collectively known as *Instant Messaging and Presence*. The IETF, being the main standardization body in this domain, has adopted two competing protocol suites in parallel, namely, *SIP presence* [2] and *Extensible Messaging and Presence Protocol* (XMPP) [3]. As presence information is event-based and generated in an ad-hoc fashion, it is disseminated in real-time from one source to many destinations following the publish/subscribe push-based communication model. This makes presence management implementations more complex compared to traditional client-server applications, especially due to typically large numbers of users. Additionally, presence solutions typically ship all generated presence updates from presentities to watchers in real-time, which means that, in practice, one of your contacts who frequently changes his/her presence status may drain out the battery on your smartphone. Thus, a solution for implicit presence maintenance needs **to reduce the frequency of presence status updates** while maintaining a suitable accuracy level. A reduced frequency of presence status updates, on one hand, saves the energy on both presentity and watcher phones, and on the other, greatly reduces the messaging load on presence servers which multiply each presence message received from a presentity with a number of its watchers, and deliver the resulting messages to watchers in real-time.

Non intrusive, flexible and energy-efficient maintenance of presence information is the main goal of our project Presence@FER [4] which builds a scalable rich presence platform. Rich presence extends the basic presence status with additional context information, e.g., operating mode of a user phone and characteristics of user environment such as location, mood or planned events in calendar. Our rich presence platform supports context-aware filtering of presence updates: Watchers may define specific interests related to their contact status, while presentities have the means to control the disclosure of their presence information. In addition, we have built an Android client [5] for rich presence which supports both explicit and implicit presence updates triggered by, e.g., calendar events or accelerometer readings. As our initial field trial demonstrates a potentially large frequency of rich presence updates, the proposed approach to presence inference in this paper is designed with a goal to reduce this frequency and achieve a favorable trade-off between energy and accuracy.

Presence inference may be regarded as a special light-weight case of activity recognition that has recently gained large interest from the research community [6], [7], [8]. We argue that although activity recognition is vital for a wide spectrum of applications, it represents too much of an overhead for state-of-the-art presence solution. On one hand, it consumes a lot of energy on end-user devices, and on the other, activity-related data is personal and sensitive. In addition, such detailed information is in general not required in communication services, especially not in corporate environments. Thus we propose a ”light-weight” approach to presence inference which detects only four possible presence states which are sufficient for communication purposes: available, available for texting, busy, and unavailable. Using the Nokia Mobile Data Challenge (MDC) dataset [9] which comprises detailed phone logs for 38 users collected over 8154 user-days, we identify a set of features for presence inference and show that the GSM cell identifier and ring tone status alone are good indicators of presence status for a large fraction of users. Therefore the use of energy-greedy sensors can be avoided for presence inference. Furthermore, our approach creates almost 3 times less load on presence servers than an approach which would change user presence status based on individual user actions, and has the potential to limit the energy consumption on smartphones due to presence messaging.

The main paper contributions can be summarized as follows: 1) We identify a set of features relevant to presence in accordance with state-of-the-art real-time communication services; 2) We design an automaton for modeling presence status changes based on user actions, and propose a novel approach to implicit change of user presence by grouping user actions and assigning presence status to groups of actions; 3) We apply the proposed presence model to prepare the data from the MDC data set as input for user presence inference, test a selected classifier performance, and identify good features for presence status inference which are also energy-efficient; and 4) We estimate the benefits of the proposed approach in terms on messaging load on presence servers based on the MDC data set. To our knowledge, this is the first published work which evaluates presence maintenance on smartphones based on an extensive real-world data log.

The paper is structured in the following way: Section II introduces a presence model and identifies a set of features relevant to presence. Our approach to learning and inferring user presence is outlined and evaluated in Section III. Section IV gives an overview of related work, while Section V concludes the paper and identifies directions for future work.

II. Presence model and feature selection

For the purpose of user presence status detection, we need to identify relevant actions which influence presence status of a user. We associate a user u_i with a set of actions ordered in time $\mathbf{A}_i = \{a_{i1}, a_{i2}, a_{i3}, ...\}$. When a user places a call, reads a message, or an event noted in user calendar starts, such action may change user presence state. There are two basic presence states: available and unavailable. A user is either *available* when he/she is willing and able to communicate by any means, or *unavailable* when he/she is not able to communicate, regardless of the reason. Furthermore, we identify two additional presence states relevant to our model: *available for texting* occurs when a user is able to communicate only via text messages, and *busy* indicates the user is currently occupied, e.g., in a meeting. The status *available for texting* is a special case of the status *available*, because when a user is *available*, he/she is also *available for texting*, while the opposite does not hold. The state *busy* is a subset of *unavailable*, because when a user is *busy*, he/she might be available to selected individuals or groups, e.g., for boss and important clients during working hours, or for family members in case of emergency, although the user is in general unavailable. Thus the identified presence states are represented by the following set: $\mathbf{S} = \{available, available\,for\,texting, unavailable, busy\}$.

The MDC Open Challenge data set used in our evaluation contains a raw list of presence-related actions (the total of 17,861,168 actions for 38 users) divided into the following logs:

- *call log* specifies the time of the call, call type (voice or message), SMS status, direction and anonymized phone number,

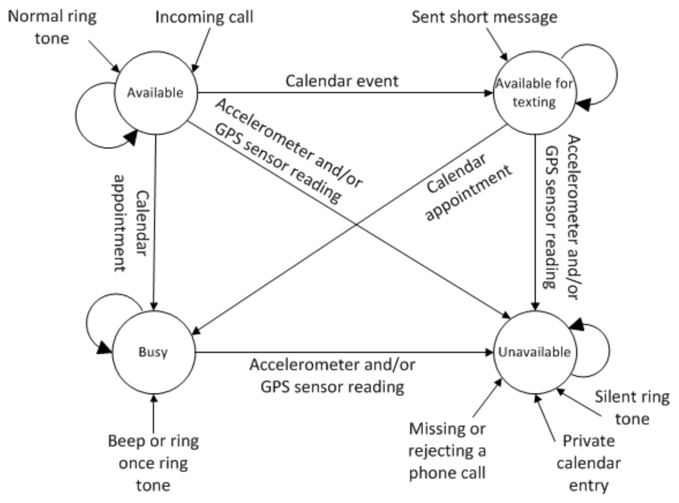

Figure 1. An automaton for user presence status assignment

- *calendar log* stores for each entry its time, status (tentative/confirmed), type (appointment/event) and class (public/private),
- *system log* identifies a phone profile (normal, silent, etc.), battery level, charging state, and ringing type,
- *accelerometer and GPS sensor log* contains accelerometer samples and GPS coordinates, and
- *GSM cell identifier log* stores anonymized cell ids.

Feature selection. The call log is vital to determine the presence status of a user because it directly indicates his/her (un)availability. We use the call type (voice call or short message) and direction (incoming, missed, or outgoing) as presence features. The calendar log is another valuable source of presence information. All calendar entries with confirmed status are relevant to presence, both appointments and events, while additional features are class (public and private) and location of the entry. The system log contains important presence-related information, such as ring type (normal, ascending, ring once, beep or silent). The accelerometer and GPS sensor logs are used in combination to classify user movement as stationary, walking, or running. Additionally, we find GSM cell identifiers related to presence because a user might be available in a particular GSM cell while unavailable in another cell.

Since the MDC dataset does not include user feedback on their presence status ("the ground truth"), we can only estimate a presence status from user actions. For example, when a user answers a phone call while the ring tone is set to normal and there are no scheduled events in the calendar, one can safely conclude that his/her status is available. Conversely, when a phone call is ignored or rejected while the user is in a meeting, it is safe to conclude the user is busy.

We define an automaton comprising the four states defined in $\mathbf{S}$, and identify significant actions which cause presence state changes. The automaton is depicted in Fig. 1 where circles represent the states while arrows indicate action-triggered transitions. Short arrows depict merged transitions from all other states to simplify the figure. For example, a user enters the state *available* when he/she answers a phone call or sets the ring tone to normal regardless of the previous state. Such actions obviously indicate user's willingness to communicate. The indicators for the status *unavailable* are the following: a missed/rejected phone call, start of a private calendar entry, and setting the ring tone to silent. Our assumption is that users do not want to be disturbed during private calendar entries and/or when they set the ring tone to silent. Furthermore, we change the presence state to *busy* when a user sets the ring tone to beep or ring once, and switch it to *available for texting* when he/she sends a short message while the previous state is either busy or unavailable. The status *available* also changes due to public calendar events and appointments. An appointment changes a user state from *available* to *busy*, while the start of a public event sets it to *available for texting*. We assume that during a public event voice conversation might be inappropriate while texting might be more convenient. Another interesting transition is triggered by accelerometer and/or GPS sensor readings which indicate that a user is traveling, running, or walking. Our assumption is that such activities might lead to the state *unavailable*. Furthermore, if a user is in a state *busy* when he/she misses a call, the automaton does not change his/her status to *unavailable*, since it is a special case of the status *unavailable*. Similarly, when a user sends a text message this action does not change his/her state from *available* to *available for texting*.

Note that the automaton changes user presence status based on individual actions (we have used this automaton to implement the rich presence Android client presented in [5]) and represents our perceived presence status management which may not hold for all smartphone users. Moreover, each action may trigger a state change which is impractical for presence service implementation due to potentially huge load on the presence server and smartphone battery limitations. Therefore, we group the actions from the set $\mathbf{A}_i$ into non-overlapping frames that contain actions from a time window of five minutes, and associate each frame with only one of the possible states. This approach is similar to the one presented in [8] for sequencing accelerometer data into 5 second frames to identify locomotive micro-activities. Our assumption is that periodic changes of user status every five minutes are acceptable for presence users while being beneficial for system load. Additionally, such approach increases the probability to correctly estimate current user status based on a sequence of activities, and produces an improved input to the classifier. Note that a comparable approach is proposed in a paper which identifies the level of intimacy in user context from the MDC data set [10]. As the MDC data set does not include the ground truth about intimacy levels, the authors have designed an algorithm to estimate it based on relevant features.

Formally, an action vector $A_{ij} \in \mathbf{A}_i$ models an action frame j for user u_i to collectively determine a user presence state. We define the following algorithm to associate A_{ij} with a state $s_{ij} \in \mathbf{S}$. First we associate each action $a_k \in A_{ij}$ with a state $s_k \in \mathbf{S}$ based on the automaton depicted in Fig. 1 to create a sequence of states S_{ij} associated with A_{ij}. Next we count the number of *available* and *unavailable* occurrences in S_{ij} (*available for texting* is counted as *available*, and *busy* as *unavailable*). In particular, we distinguish the number of *available* and *unavailable* states in S_{ij} by log types: The highest priority is given to calendar and call log entries, followed by changes of ring tone, user movements and locations determined by accelerometer/GPS sensor readings, and GSM cell identifiers. If there are calendar log entries in A_{ij} and the number of available states associated with it is larger than the number of unavailable states, we conclude the user is available. Otherwise, the user is unavailable. Finally, we decide whether the user is *available*/*available for texting*, or *unavailable*/*busy* based on the dominance of state occurrences. In case there are no calendar log entries, the same procedure is invoked on the call log, followed by the system log, etc. In the end of this process, each action vector A_{ij} for user u_i is associated with a single state s_{ij}.

III. Classification and experiments

The proposed approach to learning and inferring user presence applies the supervised learning approach which consists of a training phase followed by an online classification phase. The output of the training phase is a classification model for a single user which is subsequently used during online classification. We use the Apache Mahout API [11] in our evaluation, and have selected the Online Logistic Regression (LR) [12] as our machine learning algorithm. LR predicts a discrete outcome of an event from a set of variables that may be either numerical or categories. The Mahout implementation of LR uses the Stochastic Gradient Descent [13] optimization method, and has shown to be fast and robust for high-dimensional classification problems and large training sets [12]. LR uses action vectors $A_{i1}, A_{i2}, ...A_{in}$ associated with presence statuses $s_{i1}, s_{i2}, ...s_{in}$ as input: s_{ij} is determined for each A_{ij} based on the approach explained in Section II.

Classifier evaluation. We use a five-fold cross validation (i.e. 80-20% split of the data) to evaluate the performance of our classifier. Figure 2 shows the accuracy of the classifier for a selected subset of users which is in general extremely good for most users. We have selected seven representative users for whom the classification accuracy is more than 85% and two users with degraded accuracy (users u_9 and u_{89}). The accuracy is given for three different experimental setups: during the training phase, during online classification and evaluation including all features as classifier input (Evaluation 1), and during online classification and evaluation when a subset of features is used as classifier input. The first experiment measures accuracy and evaluates the classifier in parallel with the learning process. Evaluation 1 considers all listed features, while evaluation 2 uses only user ring tones and GSM cell identifiers as algorithm features. It is visible that the accuracy obtained during training and evaluation 1 does not deviate too much except for a few users. If a complete log from a smartphone is provided as input to the classifier, the accuracy is greater than 70% for all users, while for most users it is over 90% which is considered as an excellent result. Results obtained in evaluation 2 are more variable. However, one can conclude that ring tone and GSM cell identifiers yield discriminative information for presence inference while they are extremely favorable features in terms of energy consumption.

Table I
Classifier confusion matrix for a representative user

User 51	Available	Unavailable	Texting	Busy
Available	0.91	0.07	0.01	0.01
Unavailable	0.02	0.98	0.00	0.00
Texting	0.05	0.23	0.73	0.00
Busy	0.00	0.00	0.00	1

Table II
Classifier confusion matrix for a worst-case user

User 89	Available	Unavailable	Texting	Busy
Available	0.52	0.48	0.00	0.00
Unavailable	0.01	0.99	0.00	0.00
Busy	0.00	0.00	0.00	0.00
Texting	0.00	0.00	0.00	0.00

The following two tables investigate more deeply the accuracy with a reduced set of features. Table I shows the classification accuracy for a representative user (u_{51}) in a confusion matrix. One can conclude that the classification accuracy is quite high for states *available*, *unavailable*, and *busy*, while it is a bit lower (73%) for *available for texting*, which is typically not well represented in our training set.

Table II shows the classification accuracy for a worst-case user u_{89}. The classifier has problems to correctly classify the status available, while the accuracy is extremely high for the status unavailable. It is interesting to note that this user was not recognized to enter the states *busy* and *available for texting*, and thus the classifier has no knowledge regarding probabilities to enter those states. Our approach does require some fine-tuning to successfully cover such borderline cases.

Presence service load. Let us discuss the number of generated messages from user smartphones for implicit changes of presence status. Table III compares implicit changes of user presence when either individual preprocessed user actions or action vectors are applied. An average number of actions relevant to presence generated by 38 MDC volunteers is 16.5 per hour with a standard deviation of 9.35,

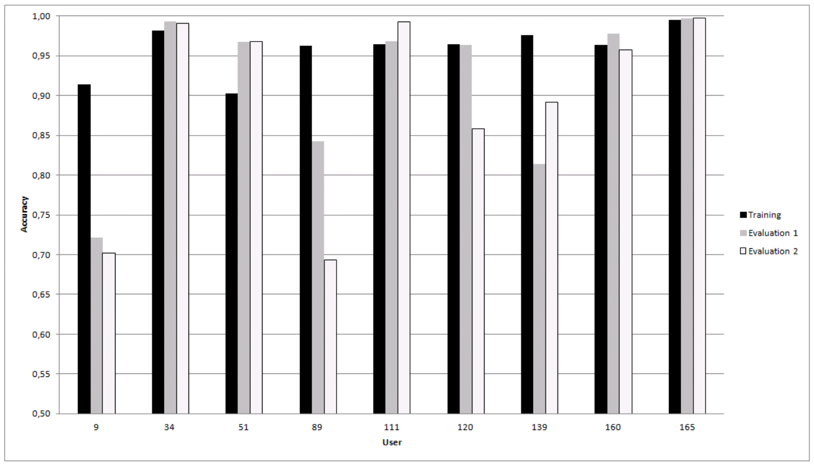

Figure 2. Classification accuracy during training and online classification

Table III
COMPARING INDIVIDUAL ACTIONS WITH ACTION VECTORS

	Average number of actions/ action vectors per user within 1 h (stdev)	Average number of state changes per user within 1 h (stdev)
Actions	16.5 (9.35)	0.72 (0.52)
Action vectors	11.4 (2.47)	0.27 (0.2)

Table IV
ESTIMATE OF PRESENCE SERVER LOAD (50.000 USERS)

	Incoming no. of messages per second	Outgoing no. of messages per second
Actions	10	200
Action vectors	3.744	74.8

while the number of action vectors within the same period is 11.4 (stdev=2.47). These actions generate on average 0.72 presence state changes within one hour for 1 user, which is on average 36,000 state changes for a average server with 50,000 users. An average number of state changes within one hour due to action vectors is 0.27 which generates the total of 13,500 implicit presence messages from 50,000 users to the presence server within one hour. Thus we conclude that our approach creates 2.67 times less incoming messages due to implicit presence status changes then the approach based on individual actions and automaton depicted in Fig. 1. In addition, as each periodic message is quite costly in wireless networks[1], potential battery savings are significant.

Table IV estimates the load on a presence server hosting 50,000 users. We compare the number of incoming server messages per second when either actions or action vectors are applied for presence updates. Again, the savings are significant, especially the number of outgoing messages which is estimated based on a conservative assumption that each user has only 20 contacts receiving his/her presence updates. Further experiments and measurements are needed to estimate true energy savings on smartphones.

IV. RELATED WORK

Presence inference is related to activity recognition which has recently gained large interest from the research community. The Nomatic prototype [6] is designed to automatically infer users' place, activity, and availability from phone sensors. The system monitors user presence over time, learns user context (place, activity, mood) from prior behavior, and prompts users to confirm their activity prior to publishing it. Thus, in contract to our approach, it requires explicit user intervention as the authors stress that presence contains sensitive personal information. CenceMe [7] supports user activity recognition from sensor-enabled mobile phones and shares this information through social networks. CenceMe designs audio and activity classifiers and presents split-level classification, whereby activity recognition (e.g., walking, in conversation, at the gym) executes in part on the phones and in part on the backend servers due to high computational requirements. In contrast to [6] and [7] that rely on multiple phone sensors to recognize user context, [8] shows that the accelerometer sensor alone provides good features to detect semantic user activities. The subsequent paper from the same group of authors investigates how the accelerometer sampling frequency and choice of features affect power consumption on smartphones [15].

We take an orthogonal approach to the previously listed works and argue that although activity recognition is vital for a wide spectrum of applications, it represents too much of an overhead for state-of-the-art presence solution. On one hand, it consumes a lot of energy on end-user devices, and on the other, activity-related data is personal and sensitive while its dissemination to others should be handled with special care. In addition, such detailed information is in gen-

[1]A recent study shows that periodic transfers in mobile application which account for only 1.7% of the overall traffic volume contribute to 30% of the total handset radio energy consumption [14].

eral not required in communication services, especially not in corporate environments. Thus we design a light-weight solution for presence inference which detects only four possible presence states which are sufficient for communication purposes. Further context information may be included in such presence update messages, e.g., user location, event type, but only with explicit user consent.

V. Conclusion

The paper proposes an approach to learning and inferring user presence on smartphones tailored to the needs of state-of-the-art real-time communication services: non-intrusiveness and energy-efficiency. We use an LR classifier from the Mahout API to perform an evaluation of our approach on the MDC dataset. The results show that the classifier achieves high accuracy with the features extracted from the call logs, calendar entries, system logs, and sensor reading from both accelerometer and GPS. Another interesting finding is that features that do not consume much energy, such as ring tone status and GSM cell identifier also provide good accuracy for presence inference. Our approach is tuned to reduce the number of presence status updates generated on smartphones which consequently greatly reduces the messaging load on presence servers as well as smartphone battery consumption due to presence messaging.

Our future work will be directed to designing a solution for presence inference on Andriod phones which can be integrated with our existing Presence@FER platform for context-aware rich presence management. This is a challenging task since it is known that traditional classifiers are not well adjusted to the limited resources of mobile devices [16]. Furthermore, we plan to conduct a small-scale study to collect explicit user input on their presence status to further evaluate our approach. This will lead us to the final goal, namely to offer automatic and energy-efficient rich presence management in mobile environments with minimal user intervention.

Acknowledgment. This work was carried out within the research project ”Content Delivery and Mobility of Users and Services in New Generation Networks”, supported by the Ministry of Science, Education and Sports of the Republic of Croatia.

References

[1] M. Day, J. Rosenberg, and H. Sugano, “A Model for Presence and Instant Messaging (RFC 2778),” IETF, February 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2778.txt

[2] J. Rosenberg, “A Presence Event Package for the Session Initiation Protocol (SIP) (RFC 3856),” IETF, August 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3856.txt

[3] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Core (RFC 6120),” IETF, March 2011. [Online]. Available: http://www.ietf.org/rfc/rfc6120.txt

[4] I. Podnar Zarko, M. Kusek, K. Pripuzic, and A. Antonic, “Presence@FER: An ecosystem for rich presence,” in *Proc. of the 11th International Conference on Telecommunications*, 2011, pp. 133–140.

[5] A. Antonic, I. Slivar, and I. Podnar Zarko, “Follow me! - A rich presence application for smartphones,” in *Software, Telecommunications and Computer Networks (SoftCOM), 2012 20th International Conference on*, 2012, pp. 1–5.

[6] D. J. Patterson, X. Ding, S. J. Kaufman, K. Liu, and A. Zaldivar, “An ecosystem for learning and using sensor-driven IM status messages,” *IEEE Pervasive Computing*, vol. 8, pp. 42–49, October 2009.

[7] E. Miluzzo, N. D. Lane, K. Fodor, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell, “Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application,” in *Proc. of the 6th ACM conference on Embedded network sensor systems (SenSys ’08)*, 2008, pp. 337–350.

[8] Z. Yan, D. Chakraborty, A. Misra, H. Jeung, and A. K., “Sammple: Detecting semantic indoor activities in practical settings using locomotive signatures,” in *Proc. 16th International Symposium on Wearable Computers*, 2012.

[9] J. K. Laurila, D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen, “The mobile data challenge: Big data for mobile computing research,” in *Proc. Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing*, 2012.

[10] M. Gustarini and W. Katarzyna, “Estimating people perception of intimacy in daily life from context data collected with their mobile phone,” in *Proc. Mobile Data Challenge by Nokia Workshop, in conjunction with Int. Conf. on Pervasive Computing*, 2012.

[11] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in Action*. Manning Publications, 2011.

[12] P. Komarek, “Logistic regression for data mining and high-dimensional classification,” Ph.D. dissertation, Carnegie Mellon University, 2004.

[13] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proc. of the 19th International Conference on Computational Statistics (COMPSTAT’2010)*. Paris, France: Springer, August 2010, pp. 177–187.

[14] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, “Periodic transfers in mobile applications: network-wide origin, impact, and optimization,” in *Proc. of the 21st international conference on World Wide Web*, ser. WWW ’12. ACM, 2012, pp. 51–60.

[15] Z. Yan, V. Subbaraju, D. Chakraborty, A. Misra, and K. Aberer, “Energy-efficient continuous activity recognition on mobile phones: An activity-adaptive approach,” in *Proc. 16th International Symposium on Wearable Computers*, 2012.

[16] D. Chu, N. D. Lane, T. T.-T. Lai, C. Pang, X. Meng, Q. Guo, F. Li, and F. Zhao, “Balancing energy, latency and accuracy for mobile sensor data classification,” in *Proc. of the 9th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys ’11. ACM, 2011, pp. 54–67.