Thyme: Improving Smartphone Prompt Timing through Activity Awareness

Samaneh Aminikhanghahi, Ramin Fallahzadeh, Matthew Sawyer, Diane J. Cook, and Lawrence B. Holder
Washington State University, Pullman, WA
s.aminikhanghahi@wsu.edu, ramin.fallahzadeh@wsu.edu
m.sawyer@wsu.edu, djcook@wsu.edu, holder@wsu.edu

***Abstract*—Smartphone prompts and notifications are popular because they provide users with timely and important information. However, they can also be an annoyance if they pop up at inopportune times and interrupt important tasks. In this paper we introduce Thyme, an intelligent notification front end that uses activity recognition and machine learning to identify the best times to prompt smartphone users. We evaluate performance of an activity-aware prompting approach based on 47 participants with fixed time and Thyme-based prompts. Our results show that responsiveness improves from 12.8% to 93.2% using this intelligent approach to timing of smartphone-based prompts.**

Keywords—activity recognition; mobile computing; notifications; user modeling.

I. INTRODUCTION

Among the valuable functions that a smartphone provides are prompts. These can be reminders or notifications that pull information related to a user's schedule, app status, or social network events and push that information to a pop-up window on the phone. Smartphone prompts are even more valuable when they are context aware, providing the user with insights corresponding to a time of day or a location. Notifications allow users to keep informed and up to date without actually firing up an app. If these notifications are in the form of prompts to perform an action, they can keep individuals on schedule and active. They can even provide a valuable type of cognitive prosthetic for individuals who require memory assistance [1]. The flip side to the convenience of phone prompts is the annoyance of receiving notifications at inopportune times. Receiving too many notifications or receiving prompts at the wrong times can increase the user's cognitive load, introduce task errors, and cause annoyance not only for the users but others nearby.

We hypothesize that smartphones can customize the timing of prompts and notifications using machine learning techniques to identify activity-sensitive contexts that are appropriate interruption points. We also postulate that adopting this activity-aware approach to prompt timing will actually improve user response rates to the prompts.

In this paper we introduce an algorithmic approach to providing activity-aware prompt timing. The algorithm is called Thyme, named after a plant that has adapted well to recent climate change [2]. Like the plant, Thyme adapts prompt times to each individual based on the user's activity context. We evaluate Thyme by using it to suggest times to ask users for information about their activities. When the user lets the query time out, the context is used to generate negative data points to a machine learning algorithm that learns appropriate prompt timings. If the user does answer the query then not only does that context provide a positive data point for Thyme's learner but the user's activity label is also used to train an activity recognizer. The activity labels in turn provide a richer source of contextual information to learn appropriate prompt times. Data collected from 47 participants indicates that appropriate prompt times can be effectively learned and providing prompts at these times dramatically increases user response rates.

II. RELATED WORK ON NOTIFICATION TIMING

Smart phone technology has become ubiquitous – the estimated number of worldwide smart phone users in 2017 is 2.32 billion [3]. With this technology comes an increase in user interruptions due to phone calls, reminders, prompts, and notifications. As a result, researchers have investigated the effect that such numerous interruptions can have on users. Certainly these interruptions can derail a user's attention from their current task [4]. They may therefore introduce task errors and result in negative psycho-physiological states [5]. Receiving too many notifications or receiving prompts at the wrong times can increase the user's perception of mental effort [6] as well as externally-observed cognitive load [7], [8].

Some mobile device users find that they can multitask easily. While this ability to quickly switch between tasks generalizes to many domains, it does degrade in situations where a person must make an effort to remember where they are in an interrupted task when they return back to it [9], [10]. Because heavy task switching is associated with reduced gray matter in the anterior cingulate cortex, there is evidence that extensive overloading of human attention may also lead to cognitive loss [11]. In addition to the effect that smart phone-based interruptions have on users, there is also a deleterious effect for app developers themselves because poorly-timed prompts reduce favorable response rates [12], decrease the ability for the user to swiftly and correctly respond to the information [13], [14], and increase the desire to turn off notifications, the app, or the device itself [15].

In response to these observations, research has been initiated to provide user support in the face of these numerous interruptions. Some of these efforts have focused on

identifying interruption situations that should specifically be avoided. For example, Gillie and Broadbent [8] have found that interruptions are particularly harmful if the user's current activity is similar to the topic of the interruption and if the interruption requires greater than average time to process. The nature of the current activity is also important: interruptions may be worse when the user is taking part in an easily distractable activity [16]. On the other hand, there is evidence that users are more accepting of notification interruptions when they have finished one task and have not yet started another [17], [18]. Furthermore, users are more welcoming of interruptions when they are at home and not engaged in other face-to-face interactions [19].

Ultimately, apps need to be designed to provide notifications at times that are best suited for the individual user [20]. Some attempts have been made to do this. Horvitz and Apacible [21] build a dynamic Bayesian network to identify amenable times for interruption based on the user's current pose and conversation level. This approach achieved accuracies as high as 0.63 based on two monitored subjects. Okoshi, et al. [22] find breakpoints between scripted activities with an accuracy of 0.83 and interrupt with task information during those times in order to reduce cognitive load [22]. Pielot, et al. [23] used phone interaction cues including screen activity, proximity, and ringer mode to predict user responsiveness to notifications with an accuracy of 0.71 for 24 instant messaging users. Smith et al. [24] compared a variety of machine learning algorithms to also predict user responsiveness based on a feature vector that included the day of week, month, time, incoming phone number, cell tower id, and WiFi SSID. After under sampling the majority class, they were able to predict responsiveness of 3 subjects over 16 weeks with an accuracy that approached 0.80.

As these previous studies highlight, mobile computing systems need to consider context to not only adapt the content of information but also the timing of the provided information. Many of these earlier efforts do consider context when predicting the response of an individual to a notification. Such context has included the time of day, day of week, and phone status. In the work described here, we propose to enhance traditional notions of context-aware systems by introducing a personalized notification timing approach that is activity-aware. Activity-aware systems utilize activity recognition to incorporate a user's behavioral patterns into the personalized system. Activity-aware systems have recently been tested for applications including operating strategy recommendation [25] and office automation [26]. In the case of Thyme, we incorporate it into a mobile app that uses awareness of a user's activity context to prompt them for information. One unique aspect of this work is the integration and evaluation of activity awareness in the timing of notifications. Another unique component is the fact that instead of predicting user responsiveness on historic data we test the ability to provide prompts at convenient times based on real-time data collection.

III. Activity Learning

One type of smartphone app that relies on user interaction is an activity learner app. Learning and understanding observed activities is at the center of many fields of study and is essential when creating apps that are sensitive to the needs of their users. An activity recognition algorithm learns a mapping from a sequence of sensor readings to a corresponding activity label. More formally, let A = {a1, a2, …, aT} be the set of all activities, where ai corresponds to the ith activity class. Given a sequence of n observed sensor readings, <r1 r2 .. rn>, a feature vector X can be extracted from the sequence. An activity recognition algorithm then learns a function h that maps the feature vector onto an activity label, $h:X \rightarrow A$. Activity recognition has been explored for a variety of sensor platforms including ambient sensors, smartphones, wearable sensors, cameras, and microphones [27], [28]. Activity labels provide a rich vocabulary for describing human behavior and a valuable way of representing contextual settings.

We have designed a mobile app, called AL [29], [30], which performs activity learning on iOS and Android mobile platforms[1]. AL collects 5 seconds of data at intervals specified by the user (fixed, custom times, or continuously). When the data is collected, AL uses its current model to identify the activity. Updated models are periodically sent to the mobile device to provide increasingly accurate activity labels.

To learn a mapping from raw sensor values to an activity label, AL needs to extract features from the raw data. Table 1 summarizes the raw sensor values that are collected from the phones and the features that are extracted from these readings. In addition to standard signal processing features, AL also extracts features that reflect higher-level information about the entire 5-second data sequence including heading change rate (percentage of points in the sequence that change directions), stop rate (percentage of points in the sequence that exhibit a significant drop in velocity), overall trajectory from start to finish of the data sequence, and normalized distance to the user's mean location.

A second input requirement for AL is a sufficient amount of labeled training data from which it can learn a model. To obtain these labels, AL periodically queries the user to ask them about their current activity. If AL guesses the label correctly the user can indicate this, otherwise the user selects an activity label from a list (or defines a new activity category). The AL interface is shown in Fig. 1.

Given enough training data instances, AL can learn activity models that map sensor data to activity labels. Many different supervised machine learning algorithms can be used to perform the actual mapping. In our experiments with alternative approaches, the alternative strategies performed comparably (the results are summarized in Section 5). For our experiments we utilize a random forest classifier. This algorithm achieves better performance than other learners on this type of activity data [29] and is computationally efficient. Our random forest implementation builds an ensemble of 100 decision trees, each of which considers a random subset of 20 features at each node in the decision tree. The resulting ensemble of trees votes on the final label for a new data point during testing. This ensemble method addresses the overfit problem that can arise when using a single decision tree classifier for high-dimensional data such as AL's phone sensor data.

[1] https://itunes.apple.com/us/app/activity-learning/id1114204788

TABLE 1. Raw data (white), features (green), and activity classes (blue).

Domain	Number	Types of Features
Raw Sensor data	16 sensor values	x / y / z acceleration, x / y / z rotation, yaw, pitch, roll, course, speed, horizontal / vertical accuracy, latitude, longitude, altitude
Phone timing data	2 values	Date, time
AL statistical features (applied to raw sensor data)	16 * 16 = 256 features	max, min, sum, mean, std, median, mean absolute deviation, median absolute deviation, zero crossings, interquartile range, coefficient of variation, skewness, kurtosis, signal energy, log signal energy, power
AL relational features (comparisons bw sensor values)	2*(3!) + 16 = 31 features	correlation (between axes for 3-dimensional sensors), auto-correlation
AL time features	4 features	auto-correlation, day of week, hour, minute, second
AL segment features	4 features	heading change rate, stop rate, trajectory, normalized distance from user location center
Thyme features	8 features	Current activity (AL), engagement level, latitude, longitude, altitude, day of week, hour, minute
User-specific activities	2 .. 25 classes	Varies by user
Generalized activities	11 classes	Bathe, Cook, Drive, Eat, Exercise, Groom, Hobby, School, Sleep, Study, Work

In addition, we enhance the random forest algorithm by making it cost sensitive. Data collected from human participants reflects the natural behavior of those participants. As a result, the number of data points in each activity class is typically far from being uniformly distributed. Because the goal of supervised learning algorithms or classifiers is to optimize prediction accuracy for the entire data set, most approaches ignore performance on the individual class labels. Therefore, a random classifier that labels all data samples from an imbalanced class dataset as members of a majority class would become the highest performing algorithm despite incorrectly classifying all minority class samples. To address this problem, during training we weight each data point with a value that is inversely proportional to the frequency of its class. These weights will guide the learning algorithm to devote sufficient attention to activity classes that occur less often. The effectiveness of cost-sensitive learning methods has been validated both theoretically [31], [32] and empirically [33], [34]. Cost-sensitive learning methods have also been coupled with existing learning methods to boost their performance as we are doing here [35].

Because of the insights that automated activity recognition offers on human behavior and the enriched context description that activity labels bring to mobile systems, activity recognition is a highly-investigated area of research [36]–[40]. Methods have been developed that encompass a diversity of sensor platforms including ambient sensors such as motion detectors, object sensors, wearable or phone sensors, and audio or video data. Researchers have utilized and enhanced many diverse learning approaches including support vector machines, Gaussian mixture models, decision trees, and probabilistic graphs [29], [36], [37], [41]–[46]. These models trade off computational cost, the type of sensor data that can be processed, and recognition performance. One distinguishing feature of AL is the ability to perform activity labeling in real time for unscripted activities. AL does not require data segmentation and has demonstrated the ability to recognize activities even in cases where activities are interrupted or performing in a multi-tasking setting [41].

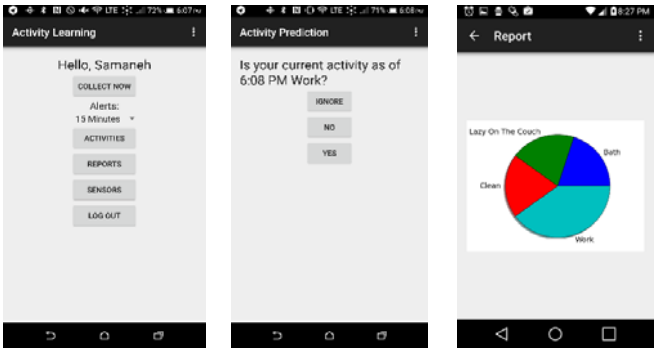

Fig. 1. AL activity learner app with user-specified activity query frequency (top), a sample query (middle), and activity reports (bottom).

AL represents a class of apps that requires interaction with its users. However, if it interrupts the user at inconvenient times, response rates will decrease and eventually the app will be stopped or removed. Our goal is thus to learn the contexts in which users are willing to be interrupted for this type of interaction. We evaluate our ability to achieve this goal based on the number of prompts that elicit a response from the user. Future studies can expand on this performance measure to include additional aspects of user experience including whether the user felt the interruption came at a convenient time, if there was another time that would be preferred, whether the notification information was understandable, and the ease with which the user could respond to the query.

We next explain our approach to accomplishing the goal of our activity-aware prompting system. In some cases, location and/or time may indicate whether a person is interruptible. For example, Jim might answer prompts when he is at home but not when he is working at his job (other than during a lunch break). Matt is interruptible when he is getting ready for bed at night but not when he is quickly getting dressed in the morning. In other situations, however, these two parameters may be insufficient to determine a person's context. Sarah may not mind her phone asking questions while she is walking alone but not when she is jogging the same route for exercise with her friends. In these cases, understanding and recognizing a person's activity can provide richer contextual information for learning valuable prompt times.

IV. THYME

Thyme is an intelligent prompting model that takes advantage of pervasive computing, signal processing, and machine learning techniques to provide personalized, activity context-aware prompt timings. Thyme uses both smartphone-based sensor features and AL-provided activity labels to gather contextual data. Thyme's machine learning algorithm maps this

input to a label indicating that the user can be prompted, "YES", or cannot be prompted, "NO", in this context.

The timing of Thyme prompts is personalized to each user. The Thyme prompting model thus follows the flow shown in Figure 2. The prompting app (in this case, AL) initially pushes queries at a fixed rate. The user can choose to answer or to ignore the query. The corresponding data represents a training instance for Thyme's learner and the user's response (or lack of response) represents the YES or NO label for the instance. Each data point with the YES/NO label is input to a decision tree learner which creates a model mapping data to corresponding labels.

The prompt content is shown in Figure 1 (middle). The user is queried about their current activity and whether it matches the label generated by AL. If the user indicates that the label is correct then the notification window disappears. If the label is incorrect then the app is brought up and provides the user with the chance to indicate the correct label for their current activity. Information about the prompt including the prompt date, prompt time, prompted activity label, and user response are stored in a relational database.

Once the model is learned, the initial fixed prompt intervals can be replaced with prompt times suggested by Thyme. Although we test Thyme prompt timing with the AL activity learner app, Thyme can be used as a prompt timing selector for any app. To use Thyme, the app such as AL continues to collect data and prepare user queries as normal. However, AL first checks with Thyme to determine if it is a good context in which to prompt or not. If it is not, the prompt is suppressed. If it is, the prompt is delivered to the user and the response can be used to refine the models for both Thyme and AL.

As Fig. 2 indicates, the activity recognition performed by the AL algorithm is used in two ways. First, AL labels the collected sensor data with corresponding activity classes and becomes part of the inferred user context in order to learn user-customized times for user interaction and prompting. Second, AL is the app which itself relies on user prompting in order to obtain sufficient labeled examples of activity categories. In theory, the learned user context model would become stronger with time as more activity labels are provided and activity recognition is more reliable. In practice for our evaluation, however, we do not update the AL models once prompting begins. This allows us to report AL accuracy that would remain fixed throughout the evaluation period. As a result, we can observe the responsiveness of the user to an activity-aware prompt strategy and discuss the likely impact of activity recognition on that responsiveness.

An activity-aware system utilizes knowledge of the current activity being performed, in addition to other contextual information such as time and location, to adapt the system and its related services. Identifying current activities providers a rich source of information for systems that can adapt to each user's behavioral patterns [47]. In addition to the current activity label, Thyme utilizes eight additional features in order to learn user-sensitive times to prompt for activity labels. The set of features are described in Table 1 and consist of the current user activity (identified by AL), the day of the week, hour of the day, number of minutes past midnight, the user's current location, and the user's recent level of engagement with the app (indicated by how quickly the user responded to the previous prompt).

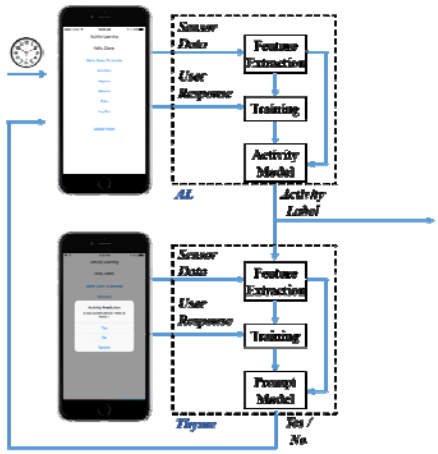

Fig. 2. An overview of the Thyme prompting model

V. Evaluation of Personalized Prompt Timings

To evaluate our Thyme-based prompt timings, we recruited n=47 adult participants to respond to queries using the AL app with and without Thyme selection of prompt times. Participants include 14 females and 33 males, ages 21-31. We collected participant data in two phases, each one week in length. The same pool of participants were used for both phases of data collection. In the first phase, AL generated activity label queries at fixed intervals of 30 minutes, 24 hours a day. We instructed participants to only respond to prompts at convenient times, otherwise they could select "Ignore" or let the query time out after 7 minutes of not responding. The AL-generated sensor data was collected, encrypted and transmitted to our server where it was stored in a password-protected SQL database together with the user response.

We used the collected data in two ways. First, the AL activity learner used features extracted from the data to learn a mapping from sensor data to the activity classes that the users defined. The first week of data is used for this purpose with the features summarized in Table 1. Second, Thyme used the response (or lack of a response) together with the AL-generated activity label to learn a mapping from the user's activity-enriched context to a decision of whether a prompt can be generated (YES) in this context or not (NO).

During the second data collection phase, we inserted Thyme into the front end of AL. The second week of data is used for this purpose. Although the prompt queries were still generated every 30 minutes, AL checked with Thyme whether it was an appropriate context for querying the user. Only in situations where Thyme output "YES" did the user receive the actual prompt. A total of 64,000 data samples were collected over the two phases. From the original forty-seven participants we created models for the users who responded to queries for the entire two weeks, or thirty-one users.

There are several aspects of the activity-aware prompt models that we evaluated in this study. First, because activity

labels are a key part of activity-aware prompt timing, we evaluate how well the AL activity models are learned for each user individually and for the whole group. Second, we evaluate how well Thyme learns prompt timing models individually and for the group as a whole. Finally, we observe the effect of Thyme-based prompt timings on user response rate for the AL app. The results indicate the usability of activity-aware prompts and notifications for apps that customize to particular users or to an entire population. The learned decision trees themselves also shed light on the features that are particularly valuable in learning activities and prompt timings for this study.

A. Evaluation of Activity Learning

To generate prompts for Thyme, we used data collected in the first phase to train AL activity models. Our participants produced a wide variety of responses both in frequency of responses and in number of defined activity classes. As Table 1 indicates, the number of activity classes ranged from 2 to 25. A total of 11,482 data instances were provided with activity labels but the number of instances for individuals ranged from 39 to 2,645. The results of Thyme-based prompting depend to a large extent on the ability of AL to correct identify an activity-based user context. We therefore evaluate AL's performance on this collected week of data in three ways, each based on 10-fold cross validation. First, we perform activity recognition separately for each user. As shown in Figure 3, AL achieves an average of 82% accuracy when training and testing is performed separately for each user. In addition to accuracy we also report an extended geometric mean, or g-mean, which is calculated as the nth root of the product of sensitivity and specificity for all classes. The last performance measure is the area under the ROC curve, or AUC value. The last two measures are particularly valuable when reporting classifier performance on imbalanced datasets. As shown in Fig. 3, AL achieved an average g-mean of 0.83 and an AUC value of 0.88.

Fig. 3. AL performance averaged over the participant group where models are learned and tested separately for each user.

We performed the same experiment with alternative classifiers and report their accuracy and g-mean values in Fig. 4. The compared classifiers include a support vector with a linear kernel (SVC Linear), a naïve Bayes (NBC), a support vector with a radial basis function kernel (SVC RBF), a k-nearest neighbor with k=3 (KNN), a decision tree (DT), and a random forest (RF). Random forest consistently performs the best, so we use this classifier for the remaining experiments.

While learning a separate activity model for each user allows prompt timings to be customized for each person, it also requires a sufficient number of training data points from each person to train the model. This constraint places the burden of training models on the user and may not be practical in all situations. For this reason, we also test the activity models for user generalizability. In this case, we collapse the 124 different activity labels we observed across the participant set into 11 more general activity classes. To do this, we merge activities into one class if they are synonyms (e.g., Bake, Cook, Cooking, Meal Preparation are merged into a single class "Cook") or one is a hypernym of another (e.g., Run, Running With Pup, Walk, and Walking Dog are merged a single class "Exercise") as defined by WordNet [48]. The final classes consist of the activities Bathe, Cook, Drive, Eat, Exercise, Groom, Hobby, School, Sleep, Study, and Work.

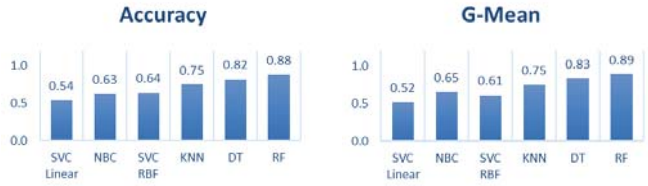

Fig. 4. AL accuracy and g-mean values by alternative classifiers, averaged over the participant group where models are learned for each user.

For our second activity recognition experiment, we test AL on this combined dataset. As shown in Fig. 5, the accuracy drops to 78%. However, the performance is significantly ($p<.05$) better than random labeling, which would yield an accuracy of 9% for this data. The results indicate how the learned model would perform for a new user even in the absence of training data for the new user.

Fig. 5. AL performance averaged over the participant group where a single model is learned for the entire collection of data.

Fig. 6. AL performance for leave-one-subject-out testing, averaged over all the participants.

For the third experiment, we test AL with for the 11 activity classes using leave-one-subject-out testing. Here the performance is quite a bit lower than situations where training data is available for the same user that is being tested, as shown in Fig. 6. The performance is still significantly ($p<.05$) better than random guess but the results of this experiment clearly indicate that some labeled data will be needed for each user to effectively learn an activity model that can be used to inform prompt timing. So, we utilize the first week of collected data to train a model for each user then collect a second week of data with Thyme-informed timings for the same set of users.

One other point we want to note is the distribution of labels across the generalized set of 11 activity classes. This distribution is shown in Fig. 7. We note that the greatest number of activity occurrences is in the Hobby category, with many fewer occurring in the Work and School categories. Interestingly, we later observe that users are more reluctant to answer queries at work or school than while they are relaxing with a hobby. This becomes a cyclic problem because notifications were used to obtain initial activity labels and these labels are used to learn prompt times. If we cannot adequately learn the activities when users should not be prompted then the prompt times will suffer.

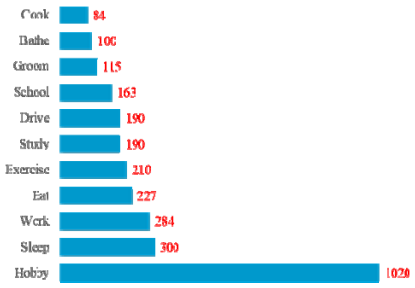

Fig. 7. Distribution of data points for the generalized activity classes.

B. *Evaluation of Prompt Timing*

Given the activity labels provided by AL and the user responses from the first phase of our experiment, we can now train the Thyme models to learn appropriate prompt times for each user. To perform this task, Thyme maps a vector of features representing a contextual situation to a YES or NO value. A value of YES indicates that a prompt can be given in the corresponding context and NO indicates that this is not a good time to prompt. Based on the nature of the training data, we can infer that a YES label also indicates a high likelihood of the user responding to the app at the prompt time, while a NO label is likely to a situation which is inappropriate or inopportune for prompting-based interruptions.

Evaluation of the learned Thyme models is based on a second week of prompting users for activity labels. During this week, prompts are only provided in contexts where the Thyme model suggests a YES value. Figs. 8-13 summarize the results of learning the prompt models. In this case we are determining whether there are patterns to user responses that can be captured by sensor data and modeled by a standard supervised learning algorithm. We first evaluate the performance based on a separate model for each user that uses the individualized AL models described in the previous section and evaluated in Fig. 3. We evaluate the models using 10-fold cross validation and the results are shown in Fig. 8.

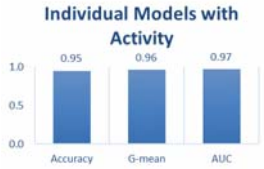

Fig. 8. Thyme performance averaged over the participant group.

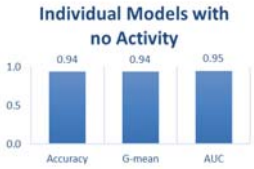

Fig. 9. Thyme performance averaged over the participants. No activity feature is used for these models.

Because the class distribution is heavily skewed we also perform minority class resampling [49]. The individual models yield an average accuracy of 95% for the thirty-one participants that generated at least one week's worth of data for each phase. Because of the heavily skewed class distribution we report g-mean and AUC values in addition to accuracy values. To determine the impact that including an activity label has on performance, we also run the experiment without this information included in the input vector. In this case the performance does degrade to 94%. The difference in performance, however, is not statistically significant.

As before, we are also interested in assessing how well a generalized model of prompt timing could perform if it is adapted to a population of individuals rather than customized for each person. In this case we use the generalized AL model for ten activities and from this information build a single Thyme prompting model. The generalized model results in a 99% accuracy with (Fig. 10) and without (Fig. 11) activity information. In this case, the difference in performance between the model that utilizes activity information and the one that does not is extremely statistically significant ($p<.01$).

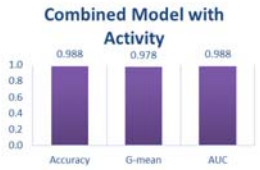

Fig. 10. Thyme performance with one model learned from data for the entire group.

Fig. 11. Thyme performance with one model learned from data for the entire group. No activity feature is used for these models.

Fig. 12. Thyme averaged performance for leave-one-subject-out testing,

Finally, we perform leave-one-subject-out testing with the Thyme-learned prompt models. As Figs. 12 and 13 show, performance is much weaker in these cases, with and without the activity information. The difference in accuracy between the model with and without activity information is not statistically significant. These results are consistent with our observations that preferred prompt times are highly individualized and need to be adapted to each person.

Fig. 13. Thyme performance for leave-one-subject-out testing, averaged over all participants. No activity information is used in these models.

Thyme utilizes many features to define context, which includes the traditional time-based context features and location-based features as well as user engagement level (how quickly the user responded to the previous prompt) and the automatically-recognized current activity. We are interested in performing feature selection to determine which contextual features provided the strongest indication that the user was interruptible and willing to respond to a prompt. For this experiment, we used a decision tree classifier to learn Thyme-based prompt contexts. We examine the rules learned by the

decision tree to see if the activity labels are critical in differentiating the classes. Decision tree learners provide this valuable feature because they select features to query in a greedy manner based on each feature's ability to discriminate between the alternative target class values [50].

In the case of the learned generalized prompting model, the user's engagement level was the most discriminating feature. This is an intuitive result because if a user is willing to respond to an earlier query that level of engagement will stay high for a period. The next most discriminating feature was the activity label. As an example, participants did not like to respond to AL queries when they were at work but were generally responsive when they were relaxing or working on a hobby. The day of the week and time of day were generally used next, followed by actual user GPS coordinates. The GPS coordinates do not generalize well to an entire population. In contrast, however, they were very discriminating when learning separate prompting models for each user.

We compare the decision tree-based feature selection method, which is based on the reduction in entropy over the dataset that is afforded by each feature, with alternative statistical methods. Specifically, mutual information and chi2 feature selection methods also rank user engagement as the top feature, followed by activity label. The remainder of the feature ordering varies between selection methods.

C. *User Response Rate*

The previous experiments are useful because they provide an indication that activities can be learned from smartphone sensor data and that prompt times can be learned as well. Smart phone sensor data provides an enough data to learn these models and there are clear rules which govern when prompts should and should not be given. In our final experiment, we measure actual user responses with fixed-time prompts (phase one) and with Thyme-based prompts (phase two). Figure 14 plots the response rate for both conditions, averaged across our set of participants. As the graph shows, Thyme generates a much more favorable response from users than fixed times. The fixed-time strategy yields an average user response rate of 12.8%. In contrast, the Thyme strategy yields an average user response rate of 93.2%. The difference in performance is extremely statistically significant ($p<0.01$).

We made several observations about the nature of the data collected in our study. First, our participants were recruited locally. As a result, their locations and activity classes were more homogeneous that we would find in the general population. In addition, because data collection occurred over a small period of time the "day of week" feature was not consistently helpful in discriminating activities or prompt times. For both of these reasons, Thyme needs to be tested for a greater diversity of users over a long period of time to ascertain its generalizability. Finally, we note that because users appeared to be less likely to respond to prompts at work, at school, or while sleeping, there were fewer data points from these categories to train AL and thus to train Thyme.

In the future, we can consider one-class learning strategies to identify prompt contexts from heavily-skewed class distributions or only from activities and contexts in which users are responsive to prompts and notifications. We also want to extend our study to evaluate the overall user experience. Feedback from users can indicate general preferences of timings for prompts and for content of prompts, both of which can provide valuable pointers for creating effective interfaces and notification timings.

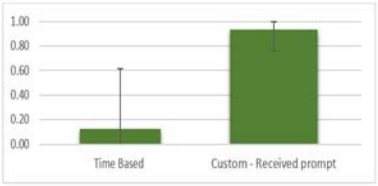

Fig. 14. Average user response rate for fixed-interval prompts and for customized prompts based on Thyme.

VI. CONCLUSION

In this paper, we introduce an activity-aware approach to customizing smartphone prompts and notifications. The proposed approach, implemented in our Thyme system, can be used to manage any smart phone notification strategy. Experimental results, obtained for a set of participants answer activity queries for a mobile app, indicate that the proposed algorithm outperforms traditional time-based prompting. The resulting app increases user response rates and decreasing inappropriate notifications. The resulting approach is useful for presenting information and interacting with users in an effective manner without creating interruption overload.

ACKNOWLEDGMENT

The authors would like to thank Bryan Minor and Larry Holder for their help with designing the AL app. This research was supported in part by NIH grant R25AG046114.

REFERENCES

[1] A. M. Seelye, M. Schmitter-Edgecombe, B. Das, and D. J. Cook, "Application of cognitive rehabilitation theory to the development of smart prompting technologies.," IEEE Rev. Biomed. Eng., vol. 5, pp. 29–44, Jan. 2012.

[2] H. Thompson, "Ten species that are evolving due to the changing climate," Smithsonian Magazine, 2014. .

[3] Statistica, "Number of smartphone users worldwide from 2014 to 2020," The STatistics Portal, 2017. .

[4] E. Horvitz and J. Apacible, "Learning and reasoning about interruption," in International Conference on Multimodal Interfaces, 2003, pp. 20–27.

[5] B. P. Bailey and J. A. Konstan, "On the need for attention award systems: Measuring effects of interruption on task performance, error rate, and affective state," J. Comput. Hum. Behav., vol. 22, no. 4, pp. 709–732, 2006.

[6] T. Okoshi, J. Ramos, H. Nozaki, J. Nakazawa, A. K. Dey, and H. Tokuda, "Reducing users' perceived mental effort due to interruptive notifications in multi-device mobile environments," in International Joint Conference on Pervasive and Ubiquitous Computing, 2015, pp. 475–486.

[7] T. Okoshi, J. Ramos, H. Nozaki, J. Nakazawa, A. Dey, and H. Tokuda, "Attelia: Reducing user's cognitive load due to interruptive notifications on smart phones," in IEEE International Conference on Pervasive Computing and Communications, 2015, pp. 96–104.

[8] T. Gillie and D. Boradbent, "What makes interruptions disruptive? A study of length, similarity, and complexity," Psychol. Res., vol. 50, no. 4, pp. 243–250, 1989.

[9] A. L. Zulas, “Modifying smart home to smart phone notifications using reinforcement learning algorithms,” Washington State University, 2017.

[10] N. Yeung and S. Monsell, “Switching between tasks of unequal familiarity: The role of stimulus-attribute and response-set selection,” J. Exp. Psychol. Hum. Percept. Perform., vol. 29, no. 2, p. 455, 2003.

[11] K. K. Loh and R. Kanai, “Higher media multi-tasking activity is associated with smaller gray-matter density in the anterior cingulate cortex,” PLoS One, vol. 9, no. 9, p. e106698, 2014.

[12] V. Pejovic and M. Musolesi, “InterruptMe: Designing Intelligent Prompting Mechanisms for Pervasive Applications,” in Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp ’14 Adjunct, 2014, pp. 897–908.

[13] J. G. Kreifeldt and M. E. McCarthy, “Interruption as a test of the user-computer interface,” in Annual Conference on Manual Control, 1981, pp. 655–667.

[14] B. Sullivan and H. Thompson, “Brain, interrupted,” The New York Times, May-2013.

[15] H. Lopez-Tovar, A. Charalambous, and J. Dowell, “Managing smartphone interruptions through adaptive modes and modulation of notifications,” in International Conference on Intelligent User Interfaces, 2015.

[16] Y. Miyata and D. A. Norman, “Psychological issues in support of multiple activities,” in User Centered System Design: New Perspectives on Human-Computer Interaction, D. A. Norman and S. W. Draper, Eds. Boca Raton, FL: CRC Press, 1986, pp. 265–284.

[17] J. E. Fischer, C. Greenhalgh, and S. Benford, “Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications,” in International Conference on Human Computer Interaction with Mobile Devices and Services, 2011, pp. 189–190.

[18] A. Exler, M. Braith, A. Schankin, and M. Beigl, “Preliminary investigations about interruptibility of smartphone users at specific place types,” in ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2016, pp. 1590–1595.

[19] G. H. ter Hofte, “Xensible interruptions from your mobile phone,” in International Conference on Human Computer Interaction with Mobile Devices and Services, 2007, pp. 178–181.

[20] M. Pielo, K. Church, and R. De Oliveira, “An in-situ study of mobile phone notifications,” in International Conference on Human-Computer Interaction with Mobile Devices and Services, 2014, pp. 233–242.

[21] E. Horvitz, P. Koch, and J. Apacible, “BusyBody: Creating and Fielding Personalized Models of the Cost of Interruption,” in Proceedings of CSCW, Conference on Computer Supported Cooperative Work, 2004, pp. 507–510.

[22] T. Okoshi, J. Ramos, H. Nozaki, J. Nakazawa, A. K. Dey, and H. Tokuda, “Attelia: Reducing user’s cognitive load due to interruptive notifications on smart phones,” in 2015 IEEE International Conference on Pervasive Computing and Communications (PerCom), 2015, pp. 96–104.

[23] M. Pielot, R. de Oliveira, H. Kwak, and N. Oliver, “Didn’t you see my message? Predicting attentiveness to mobile instant messages,” in SIGCHI Conference on Human Factors in Computing Systems, 2014, pp. 3319–3328.

[24] J. Smith, A. Lavygina, J. Ma, A. Russo, and N. Dulay, “Learning to recognise disruptive smartphone notifications,” in International Conference on Human Computer Interaction with Mobile Devices and Services, 2014, pp. 121–124.

[25] A. Doryab and J. E. Bardram, “Designing activity-aware recommender systems for operating rooms,” in Workshop on Context-Awareness in Retrieval and Recommendation, 2011, pp. 43–46.

[26] P. Jaramillo and O. Amft, “Improving energy efficiency through activity-aware control of office appliances using proximity sensing - a real-life study,” in IEEE International Conference on Pervasive Computing and Communications, 2013, pp. 664–669.

[27] D. J. Cook and Narayanan C. Krishnan, Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data. Wiley, 2015.

[28] L. Chen, J. Hoey, C. D. Nugent, D. J. Cook, and Z. Yu, “Sensor-based activity recognition,” IEEE Trans. Syst. Man, Cybern. Part C Appl. Rev., vol. 42, no. 6, pp. 790–808, 2012.

[29] K. D. Feuz and D. J. Cook, “Modeling skewed class distributions by reshaping the concept space,” in National Conference on Artificial Intelligence, 2017, p. under review.

[30] K. Bouchard, L. Holder, and D. J. Cook, “Extracting generalizable spatial features from smart phone datasets,” in AAAI Conference on Artificial Intelligence, 2016.

[31] C. Elkan, “The foundations of cost-sensitive learning,” in International Joint Conference on Artificial Intelligence, 2011, pp. 973–978.

[32] M. Maloof, “Learning when data sets are imbalanced and when costs and unequal and unknown,” in ICML Workshop on Learning from Imbalanced Data Sets II, 2003.

[33] K. McCarthy, B. Zabar, and G. Weiss, “Dost cost-sensitive learning beat sampling for classifying rare classes?,” in Workshop on Utility-Based Data Mining, 2005, pp. 69–77.

[34] X. Liu and Z. Zhou, “The influence of class imbalance on cost-sensitive learning: An empirical study,” in International Conference on Data Mining, 2006, pp. 970–974.

[35] F. Provost, T. Fawcett, and R. Kohavi, “The case against accuracy estimation for comparing induction algorithms,” in International Conferenceon Machine Learning, 1998, p. 445.

[36] A. Bulling, U. Blanke, and B. Schiele, “A tutorial on human activity recognition using body-worn inertial sensors,” ACM Comput. Surv., vol. 46, no. 3, pp. 107–140, 2014.

[37] O. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” IEEE Commun. Surv. Tutorials, vol. 15, no. 3, pp. 1192–1209, 2013.

[38] N. Yala, B. Fergani, and A. Fleury, “Feature extraction for human activity recognition on streaming data,” in International Symposium on Innovations in Intelligence Systems and Applications, 2015, pp. 1–6.

[39] Y. Zheng, W.-K. Wong, X. Guan, and S. Trost, “Physical activity recognition from accelerometer data using a multi-scale ensemble method,” in Innovative Applications of Artificial Intelligence Conference, 2013, pp. 1575–1581.

[40] J.-H. Hong, J. Ramos, and A. K. Dey, “Toward personalized activity recognition systems with a semipopulation approach,” IEEE Trans. Human-Machine Syst., vol. 46, no. 1, pp. 101–112, 2016.

[41] N. C. Krishnan and D. J. Cook, “Activity recognition on streaming sensor data,” Pervasive Mob. Comput., vol. 10, pp. 138–154, Feb. 2014.

[42] T. Barger, D. Brown, and M. Alwan, “Health status monitoring through analysis of behavioral patterns,” IEEE Trans. Syst. Man, Cybern. Part A, vol. 35, no. 1, pp. 22–27, 2005.

[43] J. K. Aggarwal and M. S. Ryoo, “Human activity analysis: A review,” ACM Comput. Surv., vol. 43, no. 3, pp. 1–47, 2011.

[44] S.-R. Ke, H. L. U. Thuc, Y.-J. Lee, J.-N. Hwang, J.-H. Yoo, and K.-H. Choi, “A review on video-based human activity recognition,” Computers, vol. 2, no. 2, pp. 88–131, 2013.

[45] A. Reiss, D. Stricker, and G. Hendeby, “Towards robust activity recognition for everyday life: Methods and evaluation,” in Pervasive Computing Technologies for Healthcare, 2013, pp. 25–32.

[46] S. Vishwakarma and A. Agrawal, “A survey on activity recognition and behavior understanding in video surveillance,” Vis. Comput., vol. 29, no. 10, pp. 983–1009, 2013.

[47] R. Fisher and R. Simmons, “Smartphone interruptibility using density-weighted uncertainty sampling with reinforcement learning,” in International Conference on Machine Learning and Applications, 2011, pp. 436–441.

[48] G. A. Miller, “WordNet: A lexical database for English,” Commun. ACM, vol. 38, no. 11, pp. 39–41, 1995.

[49] N. Chawla, K. Bowyer, L. Hall, and W. P. Kegelmery, “SMOTE: Synthetic minority over-sampling technique,” J. Artif. Intell. Res., vol. 16, pp. 321–357, 2002.

[50] J. R. Quinlan and R. L. Rivest, “Inferring Decision Trees Using the Minimum Description Length Principle,” Inf. Comput., vol. 80, pp. 227–248, 1989.