NotifyMeHere: Intelligent Notification Delivery in Multi-Device Environments

Abhinav Mehrotra
University College London, UK
a.mehrotra@ucl.ac.uk

Robert Hendley
University of Birmingham, UK
r.j.hendley@bham.ac.uk

Mirco Musolesi
University College London, UK
The Alan Turing Institute, UK
m.musolesi@ucl.ac.uk

ABSTRACT

Personal interactions and information access are happening more and more through the mediation of computing devices of various types all around us. In our daily life we use many computing devices running different versions of the same application such as email clients or social media platforms, which alert users about a new piece of information or event on all devices. In this paper we first present a study investigating the factors influencing users' decisions in handling notifications in a multi-device environment. We collected 57,242 *in-the-wild* notifications from 24 users over a period of 21 days. We found that users' decisions in handling notifications are impacted by their physical activity, location, network connectivity, application category and the device used for handling the previous notification. Finally, we show that an individualized model can predict the device on which the user will handle a notification in the given context with 82% specificity and 91% sensitivity.

CCS CONCEPTS

• **Human-centered computing** → *HCI design and evaluation methods*; *Empirical studies in ubiquitous and mobile computing*;

KEYWORDS

Mobile Sensing, Multi-device Notifications, Interruptibility, Context-aware Computing

ACM Reference Format:
Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. 2019. NotifyMeHere: Intelligent Notification Delivery in Multi-Device Environments. In *2019 Conference on Human Information Interaction and Retrieval (CHIIR '19), March 10–14, 2019, Glasgow, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3295750.3298932

1 INTRODUCTION

Cross-platform applications leverage notifications to trigger real-time alerts for steering users' attention towards newly available information through auditive, visual and haptic signals. Indeed, these notifications are beneficial to users for proactive personalized information delivery about a variety of events such as travel update information or a new comment on a social network post [8, 43]. However, notifications sometimes arrive at inappropriate moments and, for this reason, they can have an adverse impact on the execution of the ongoing tasks [7, 10, 11] and even on the affective state of users [5, 6]. The problem is exacerbated by the fact that cross-platform applications prompt users on *multiple devices at the same time*, which makes these notifications even more disruptive and annoying [39]. In light of this, a recent study shows that users prefer to receive notifications on specific devices based on their situation [42].

Until now, previous interruptibility management studies have used notification and contextual information to infer opportune moments for delivering mobile notifications (e.g., [19, 21, 25, 35, 36, 40]) and to learn the types of notifications users prefer to receive in different situations (e.g., [22, 30]). In this work, to the best of our knowledge, for the first time we try to address the problem of delivering notifications *on the right device* in different contexts. It is very difficult to define the concept of *right* device, but as a first approximation, it can be seen as the device on which users prefer to handle a specific notification given their context.

In order to design a solution for intelligent cross-platform notification delivery, we first conduct an in-situ study to investigate the contextual factors impacting users' decisions for handling notifications on specific devices. In particular, in this paper we investigate how users behave in different contexts when they receive a notification on their mobile phone as well as on a generic alternative device (i.e., any alternative computing device that runs a version of applications that are also available for mobile phones) at the same time. Through a passive logging smartphone application we collected 57,242 in-the-wild notifications from 24 users over a period of 21 days. Out of these notifications, around 44.8% of them (i.e., 25,677) were triggered by cross-platform applications. We found that 28.59% of notifications that are triggered by cross-platform applications are handled on an alternative device (i.e., not handled on the mobile phone). Our results show that users' decisions for handling notifications on a mobile or an alternative device are associated with their physical activity, location, application category and network connectivity.

Starting from this analysis, in this paper we present the design, implementation and evaluation of NotifyMeHere, a solution for intelligent notification delivery in multi-device environments. The design is based on a prediction component based on a model of user interaction with notifications on a mobile or an alternative device in different sensed contexts. NotifyMeHere can be implemented as a cross-platform application's server component that is aware of all devices of each user that have subscription to the application.

CHIIR '19, March 10–14, 2019, Glasgow, United Kingdom

ACM ISBN 978-1-4503-6025-8/19/03…$15.00
https://doi.org/10.1145/3295750.3298932

In order to assess NotifyMeHere, we implemented and evaluated predictors by using a series of machine learning algorithms including AdaBoost [34], Random Forest [9] and Recursive Partitioning [17]. Our results show that predictors are more accurate when they are trained using data of each user (with 75% specificity and 90% sensitivity) compared to generic predictors trained on data related to a set of users. In fact, the latter achieves only 37% specificity and 90% sensitivity.

2 RELATED WORK

The effects of interruptions caused by notifications occurring at inopportune moments have been investigated thoroughly in the past (see, for example, [10, 11]). Studies have demonstrated that interrupting users engaged in tasks has considerable negative impact on task completion time [10, 11], error rate [7], and affective state [5, 6].

In order to address these issues, many previous mobile interruptibility studies have investigated how users perceive notifications [31, 38] and the factors involved in users' receptivity [14, 16, 23, 26, 27]. At the same time, the advances in the area of machine learning have enabled us to model users' interruptibility by exploiting their historical notification interaction data [24]. Previous studies have proposed various mechanisms for modeling interruptibility that can be used by applications to deliver notifications at the inferred opportune moments by means of users' contextual information [15, 18–20, 29, 35, 37], the notification content [22, 25] or the prospective information [21]. In a seminal paper [19] Ho and Intille suggested that the transition between two physical activities (such as sitting and walking) can be used as an opportune moment for delivering notifications. Similarly, Fischer et al. [15] showed that users react faster to notifications that are delivered immediately at the end of a task, such as after finishing a phone call or reading a text message. In [37] Rosenthal et al. proposed a mechanism that relies on experience sampling to learn users' preferences for ringer mode settings and to predict when to mute a phone to prevent interruptions caused by notifications. Pielot et al. [32] proposed a model that can predict whether a user will view a notification within a few minutes with a precision of approximately 81%. Pejovic and Musolesi [29] proposed a mechanism that relies on contextual information (including activity, location and time of the day) to predict opportune moments for delivering notifications. Mehrotra et al. [25] designed a mechanism that relies on both contextual information and the notification content for modeling interruptibility. In a further study, Mehrotra et al. [22] proposed an interruptibility management mechanism that learns users' preferences for receiving notifications based on the automatic extraction of rules by mining their interaction with mobile notifications. The goal of their system is to be intelligible to users: for this reason, it makes the discovered rules transparent so that users can decide whether to accept or discard them at run-time.

3 DATA COLLECTION

To investigate the factors that determine users' decisions about handling notifications on specific devices in different situations, we conducted a field study to collect data through the real-world deployment of an Android app. As shown in Table 1, the app passively collects information about *in-the-wild* notification handling and context information (including physical activity, location and network connectivity).

Group	Features
Notification	Arrival time, removal time and sender application name.
Context	Physical activity, location and network connectivity.
Phone Usage	Foreground application (including home screen) usage, lock/unlock, screen change and click events with timestamp.

Table 1: Description of features from the dataset.

Our application relies on Android's Notification Listener Service [2] to log users' interaction with notifications and uses Google's Activity Recognition API [3] to obtain the information about user's physical activities classified as *walking*, *bicycling*, *commuting in-vehicle* or *still*. The application samples GPS data in an adaptive sensing fashion as described in [25]. Furthermore, the application relies on the NetworkInfo API [1] to get notified in real-time about network connectivity changes.

The Android application also collected additional data about other phone's and contextual attributes, such as ringer mode, call and SMS logs. However, we do not discuss those aspects of the data because they are not used for the analysis presented in this paper.

3.1 Recruitment of the Participants

We published the application on the Google Play Store and made it available to the general public for free. We advertised the application through different channels: academic mailing lists, Twitter, Facebook and Reddit.

In order to ensure privacy compliance, our application goes through a two-stage user agreement to access user's critical data. Firstly, a user has to give explicit permission as required by the Android operating system for capturing application usage, notifications and user's interaction with the mobile phone (such as clicks and screen changes). Secondly, the application shows a list of information that is collected and asks for user consent. It is worth noting that the study was done in accordance with our institution's ethical research procedure and the consent form itself for the data collection was reviewed by its Ethics Review Board.

4 DATASET

From 4th January to 1st June 2016, our application was installed by 87 participants from the Google Play Store. However, many users did not keep the application running actively and/or uninstalled it after a few days. Therefore, we selected a subset of the data for the analysis by considering only the users who kept the application running for at least 21 days. There are 26 users who satisfied this constraint. We also found that these users contribute minimum 80 notifications for each day. Note that we do not have information about the demographics of these participants because it was not asked during the study for privacy reasons following our institution's Ethics Board review. Our final dataset (i.e., the subset of active users) comprises 57,242 notification instances, 8.19 million phone usage events and 1.9 million context samples.

4.1 Identifying Devices on Which Notifications are Handled

First of all, we would like to underline the fact that in this work we predict the device on which a notification will be handled by the user but for only two classes of devices: personal phones and alternative devices (i.e., any device other than their phone). However, the proposed model is highly generalizable to the case of multiple classes of devices.

In order to infer whether a notification is handled or not (i.e., handled on some other device), we assume that a notification is automatically removed from the notification bar (or from the lock screen) of the phone if it was delivered on some other device and the user has already interacted with it on other device. More specifically, our assumption is that a notification is handled on a mobile phone only if the user has interacted with the phone between its arrival time (t_a) and its removal time (t_r). If there was no phone interaction logged in the interval between the time t_a and t_r, we assume that the user handled the notification on an alternative device. In fact, for instance, if a user receives a travel update notification on her phone as well as laptop, but she views and clicks the notification on her laptop then the corresponding application running on the phone automatically removes the notification from the notification bar. Therefore, to check the user's interaction during the interval between t_a and t_r, we use the phone usage data collected by our application (see Table 1). As discussed earlier, this data contains the logs for clicks, changes in the foreground and lock/unlock events.

4.1.1 Issues with Long Response Time. A recent study on mobile notifications [25] pointed out that some applications automatically kill the notifications that they generated and are not handled by the user within 30 minutes from their arrival time. This shows that some notifications can be automatically removed from the phone without being handled on an alternative device and, therefore, the notification response labels might not be correct in these cases. For this reason, we check notifications of which apps never have the value of $t_r - t_a$ more than 30 *min* and filter out all notifications of these apps that have $t_r - t_a = 30$ *mins*.

4.1.2 Removing Notifications of Active Apps. Some apps such as instant messengers could generate continuous notifications even when the corresponding app is already in foreground (i.e., actively in use). However, such notifications are not of interest for us as it is very likely that users will handle these notifications on the same device on which they are interacting with the corresponding app. Therefore, on arrival of a notification the data collection app checked whether the corresponding app is in use or not, and in case the app is in foreground the response time of the notification is set to 0. In the analysis for this study we discarded all these notifications.

4.1.3 Removing Stacked Notifications. Another issue with the data is the duplicate entries introduced by "stacked" notifications (i.e., multiple notifications from the same applications getting displayed in the notification bar). In the Android operating system, these stacked notifications are aggregated into a single notification presenting the summary of all stacked notifications. However, fortunately, the Android Notification Listener API triggers a new notification event for stack notification update and, thus, our application collected them as separate notifications.

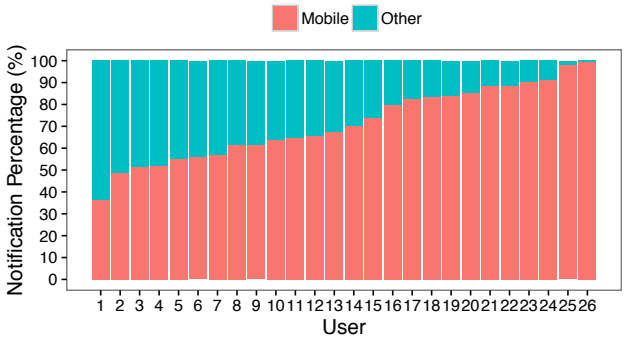

Figure 1: Percentage of notifications handled on mobile and other devices by the users.

To find the stacked notifications we iterate through each notification and check if any other notification from the same application arrived between t_a and t_r of another notification. Since all notifications stacked together receive the same response (i.e., click or dismiss) at the same time, in a sense they represent duplicate entries and are not useful for quantifying users' reactions in that specific context. Therefore, from each group of stacked notifications we keep the notification that arrived first and drop out the remaining ones.

4.1.4 Notification Distribution. In Figure 1 we show the percentage of notifications that are handled on mobile phones and alternative devices by each user. The results show that a few users handle 50% or above of the overall notifications on alternative devices, whereas a few users interact with nearly all notifications through their mobile phones. Overall, there are 71.41% notifications that are handled on mobile phones and the remaining on alternative devices.

For the evaluation of NotifyMeHere, we consider only users that handle notifications on multiple devices. Indeed, the limit (and, in a sense, trivial) case of a user interacting with a single device can be addressed simply by delivering all the notifications only on it without any prediction. Therefore, for fairness, we discard two users (users 25 and 26 in Figure 1) as they handle almost all the notifications (close to 100%) on the phone. As a result, we only consider the 24 users who satisfy this constraint. Finally, the dataset contains 69.18% notifications that are handled on mobile phones.

5 IMPACT ON NOTIFICATION HANDLING BEHAVIOR IN A MULTI-DEVICE ENVIRONMENT

In this section we investigate the effect of different factors on users' behavior with respect to notification handling in a multi-device environment.

Feature	χ^2	p-value
Activity	163.05	<0.005
Location	694.55	<0.005
Time Interval	6.14	0.1047
Network Connectivity	158.64	<0.005
App Category	792.93	<0.005
Previous Click	6465.40	<0.005

Table 2: Analysis of factors influencing users' decision for handling cross-platform notifications.

In order to perform this analysis, we link notifications to contextual and notification-based features that include:

Activity: physical activity as inferred through phone sensors. As discussed earlier, the application relies on the Google Activity Recognition API to log the information about users' physical activities classified as *walking*, *cycling*, *commuting in-vehicle* or *still*. However, we merge *walking* and *cycling* together as the *moving* activity, because we assume that while moving (i.e., either walking or cycling) users interact only with their phone rather than other devices such as their laptop. We are aware that it is unlikely that users interact with the mobile phone while biking. However, in case they want to interact with notifications in such a context, it is very likely that they will use a mobile phone rather than other devices (such as a laptop).

Location: semantic location (e.g., home, work, other). In order to cluster the GPS data we apply the clustering algorithm presented in [41]. For each clustered location we assign one of the following labels: *home*, *work* or *other*. We assign the *home* label to the place where a user spends the majority of the night hours (from 20:00 to 08:00). We generically refer to *work* as the second most significant place (i.e., the place where users spend most of their time apart from home). We are aware that this generic label might not be correct in some cases. The labeling might be improved by using geo-spatial data, but this is outside the scope of this paper. All other places are labeled as *other*.

Time Interval: time at which the notification arrived. We split a day into six time intervals: early morning (05:00-08:00), morning (08:00-12:00), afternoon (12:00-16:00), evening (16:00-20:00), night (20:00-23:00) and late night (23:00-05:00).

Network Connectivity: type of network (i.e., mobile data, WiFi or none) the phone is connected to.

Previous Click: the device on which the previous notification was handled (i.e., mobile or other).

App Category: the category of app that triggered the notification. Since the categories defined by the Google Play Store are too generic, we manually categorize the apps. Here we consider only the cross-platform applications and our categories include: *chat*, *email* and *social*.

We compute the effect of each of the above contextual and notification-based metrics (i.e., independent variables or IVs) on the user's behavior in terms of handling notifications in a multi-device environment (i.e., dependent variable or DV). Since they contain categorical values, we use Pearson's Chi-squared test [28] to investigate if there is a statistically significant relationship between the DV and IVs. In Table 2 we present the results of our analysis. We observe that there is a significant relationship between the user's behavior in terms of handling notifications and several factors, including activity, location, network connectivity, application category and the device used for attending to the previous notification. However, there is no significant impact of the time interval on the user's notification handling behavior.

The association with users' activity can be due to the fact that when users are moving they usually do not carry and interact with multiple devices. Indeed, there is the possibility that interactions with multiple devices happen while users travel for example on public transit (e.g., a train). Similarly, the association with location indicates that people do not carry multiple devices everywhere. For instance, we might carry multiple devices while at home, workplace or in a coffee shop, but not for example inside a gym. At the same time, the association with the app category indicates that people tend to handle some types of notifications on specific devices. For instance, we tend to chat more on phones but read emails more frequently on laptops due to usability factors.

6 PREDICTING THE RIGHT DEVICE ON WHICH TO DELIVER NOTIFICATIONS

In this section we discuss the implementation and evaluation of our proposed approach for predicting the device on which a notification will be handled by a user. We build two types of prediction models: (i) *individualized models* that are trained with the portion of data of a single user, i.e., the owner of the device, and (ii) *generic models* that are trained with the entire dataset containing data about all users.

As discussed above, users' notification handling behavior is not influenced by all features. Therefore, we build a prediction model that exploits only a subset of features. In particular, we only use activity, location, network connectivity, application category and the device on which the previous notification was acted upon for constructing the prediction model.

Both of these models are built by using three machine learning algorithms: AdaBoost [34], Random Forest [9] and Recursive Partitioning [17]. We evaluate these models for predicting the device on which the user will handle a notification in different contexts by using the k-fold cross validation approach with the value of k as 10. Moreover, we use the default parameters for all models available in the *scikit-learn* package [4].

In order to assess the performance of these models, we chose to compare the predicted response with the actual response (i.e., the ground truth) and compute the accuracy in terms of:

- *Sensitivity*: ratio between the number of notifications that are correctly predicted as handled on the mobile device and the total number of notifications that are actually handled on it.
- *Specificity*: ratio between the number of notifications that are correctly predicted as handled on an alternative device and the total number of notifications that are actually handled on an alternative device.

It is worth noting that we use the 10-fold cross validation approach for computing the mean and the standard-error with a 95% confidence interval of sensitivity and specificity for both models.

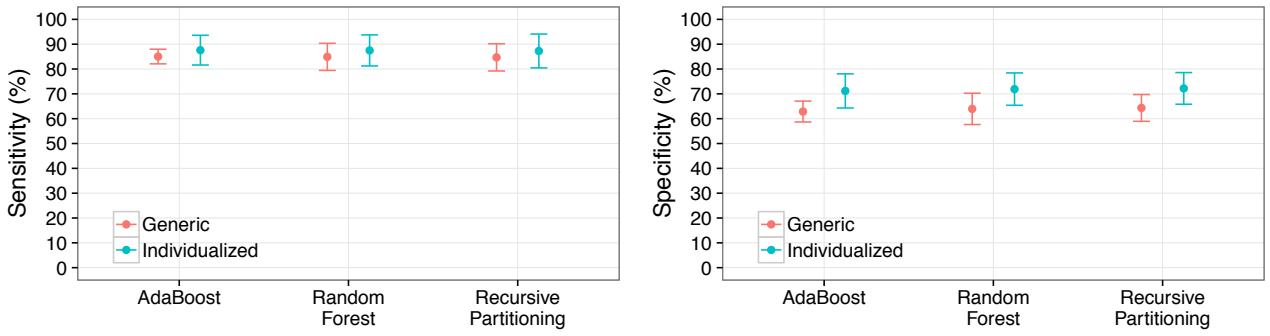

Figure 2: Prediction results for individualized and generic models using three different machine learning algorithms.

In Figure 2 we present the prediction results for individualized and generic models. The results show that for all machine learning algorithms the performance of both individual-based and generic models is similar. Both of these models achieve 85-89% sensitivity and 65-71% specificity.

6.1 Why are the Performance Results of the Individualized and Generic Models Similar?

In order to investigate why there are small differences in the performance of individualized and generic models, we explore the *importance* (based on the Gini index [9]) for each of the independent variables. It is worth noting that studies have demonstrated that the Gini index and Information Gain are the best metrics for the selection of features and there are no significant differences in their performances [33]. We opted for the Gini index as it is used internally in many classifiers, including those used in our analysis.

As shown in Table 3, the Gini index of the variable *previous click* dominates over the others (i.e., more than 20 times bigger compared to the other variables). In other words, this indicates that the *previous click* is the most important feature for the prediction of the device on which the next notification will be handled. This is due to the fact that most of the notifications are handled on the same device as their preceding notification and, thus, both models can achieve high accuracy by using solely the "previous click" feature.

Feature	Gini index
Activity	51.31
Location	79.57
Network Connectivity	21.31
App Category	89.49
Previous Click	1610.11

Table 3: Gini index of the features used for training prediction models.

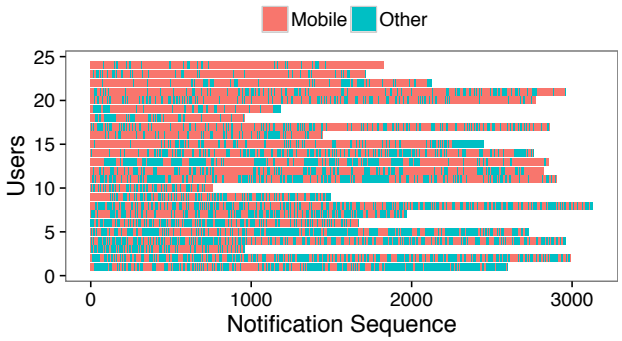

Figure 3: Sequence of notifications handled by users on mobile and other devices.

6.2 Analyzing Notification Handling Sequence

In order to understand users' behavior for handling notifications over time and to confirm that most of the notifications are handled on the same device as their preceding notification, we plot the sequence of notifications along with the device on which they are handled. As shown in Figure 3, most users handle a sequence of notifications on the same device. This demonstrates that once individuals start handling notifications on a device they keep on handling the subsequent notifications on the same device for a certain amount of time. Indeed, users do not switch devices very frequently; rather, they have a fairly prolonged interaction with a single device, especially in the case of mobile phones [13]. For instance, if a user starts chatting with a friend on her laptop then with a high probability they will continue to chat on that device instead of switching to their phone.

6.3 Impact of Time Elapsed between Subsequent Notifications

The findings presented above demonstrate that the *previous click* feature is an important variable that can solely predict the device on which the notification will be handled. This is probably due to the fact that it is very likely that a user will act upon the current notification on the same device on which the previous one was handled. However, the temporal dimension (i.e., the time elapsed between the last and current notification) might also play a key

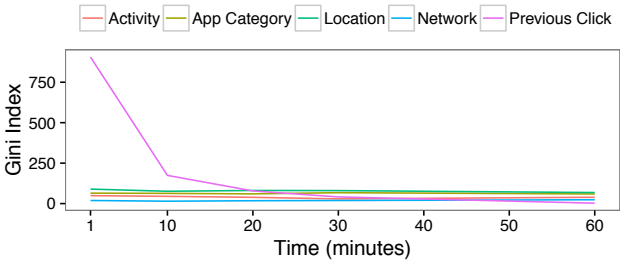

Figure 4: Importance of variables for predicting the right device.

role here since a user's context may change after some time. Indeed, after a certain time, when users move to a new environment, they might want to receive notifications on a different device that is now available to them. A recent study has shown that users' preferences for receiving notifications on a specific device changes when they switch their context [42]. Therefore, it is very important to investigate a critical question: *is the previous click feature still sufficiently informative if there is a long time between two consecutive notifications*?

For this reason, we investigate the optimal value of the time elapse (t_{elapse}) until which the *previous click* feature can be solely used for predicting the device on which a notification will be handled. Here, t_{elapse} refers to the time difference between the arrival of the last and current notifications.

In order to investigate the optimal value of t_{elapse}, we filter out the notifications with the value of t_{elapse} greater than a certain threshold ($t_{threshold}$) and compute the importance of all variables based on the Gini index. To better understand this filtering, let us consider an example with five notifications (n_1, n_2, n_3, n_4 and n_5) arriving at times t, $t + 20s$, $t + 70s$, $t + 100s$, and $t + 200s$. Now, if we want to filter notifications with $t_{threshold}$ equal to 1 minute, we would be left only with notifications n_1, n_3 and n_5. The notification n_2 is filtered out as the time gap between n_1 and n_2 is 20s (which is less than $t_{threshold}$). However, n_3 is not filtered because the time gap between n_1 and n_3 is 70s (here n_1 is considered as a preceding notification because n_2 is discarded). Similarly, n_4 is dropped as the time gap between n_3 and n_4 is below $t_{threshold}$. It is worth noting that we use the k-fold cross validation approach (with k equal to 10) while computing the variable importance to ensure that the variable importance is computed only with the training data.

Finally, we compute the variable importance by setting the value of $t_{threshold} \in [1, 10, 20, 10, 30, 60]$ minutes. As shown in Figure 4, the *previous click* feature dominates over all other features only for the $t_{threshold}$ of up to 10 minutes. *Therefore, we can conclude that the previous click feature can be used solely for predicting the device for delivering notifications but only when the preceding notification has arrived approximately within 10 minutes.* We observe that this result is essentially valid for this set of users (i.e., we are not making any claim about its universality), but we believe that the methodology described above can be applied to any population in order to estimate the optimal threshold.

6.4 Modeling Context Change

We now discuss how the performance of our model can be improved by introducing additional temporal-based features. Since the *previous click* feature is not reliable for predicting the device on which a user will receive a notification after 10 minutes, we introduce a new feature, namely *time elapsed - notification* (since the last notification).

At the same time, we believe that users' preferences might not change immediately after the context switch. Instead they would take some time to adjust to the new context and thus, their preferences would change after a certain adjustment period. Consider for example a scenario where Alice switches context between "commute" and "workplace". She uses a mobile phone during the commute and on reaching the workplace she might start using her laptop/desktop and want to get notifications on that device. However, she would not immediately start using those devices, rather she might still be receptive to notifications on her mobile phone for some time even after the context switch. Therefore, it is of fundamental importance for the prediction model to have the knowledge about when a context change happened (and, most critically, how long ago).

To provide such information to the prediction model, we introduce three more features that inform the model about the time elapsed since the switch in each context modality (i.e., activity, location and network connectivity). We name these features *time elapsed - activity, time elapsed - location* and *time elapsed - network* for the time passed since the previous switch in activity, location and network connectivity, respectively. These features can be exploited to learn how long it takes for users to change their notification handling behavior once they switch to a new context. For instance, in the example above, Alice reaches her office (which corresponds to a change in activity, location and network connectivity) but she might not start using her laptop until after a certain amount of time.

6.5 Exploiting Context Change Features to Improve the Prediction Models

We build both individualized and generic models by using the same three machine learning algorithms: AdaBoost, Random Forest and Recursive Partitioning. All models are built using the following features: *activity, location, network connectivity, app category, previous click, time elapse - activity, time elapse - location, time elapse - network* and *time elapse - notification.*

In Figure 5 we present the prediction results for both individualized and generic models. The results show that for all machine learning algorithms the individual-based models outperform the generic models in terms of specificity, but there is no significant difference in terms of sensitivity. The individual-based models achieve around 91% sensitivity and 82% specificity. However, the generic models could obtain only 90% sensitivity and 68% specificity. Overall, all models built by using time-dependent features could successfully outperform the frequency-based model that we use as the benchmark.

These results demonstrate that by introducing new features both the sensitivity and specificity of the individualized models improve by 2% and 11% respectively. On the other hand, the sensitivity and

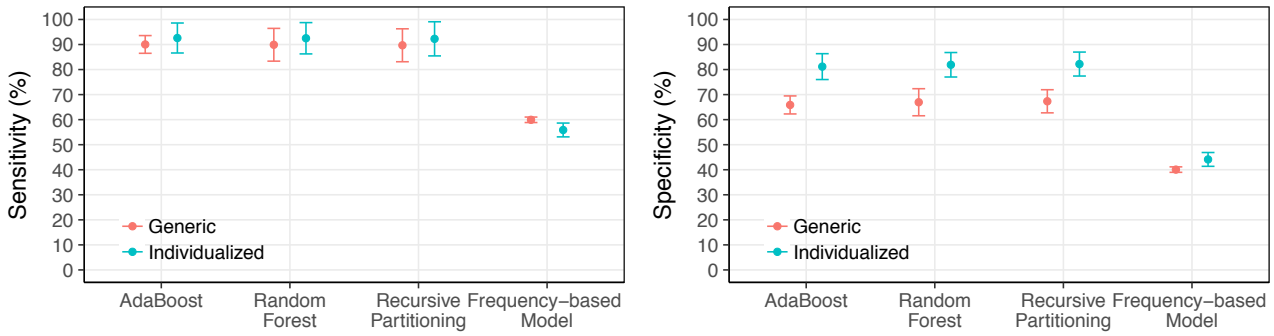

Figure 5: Prediction results for individualized and generic models built on three different machine learning algorithms by using time-dependent features and the frequency-based model (used as a baseline).

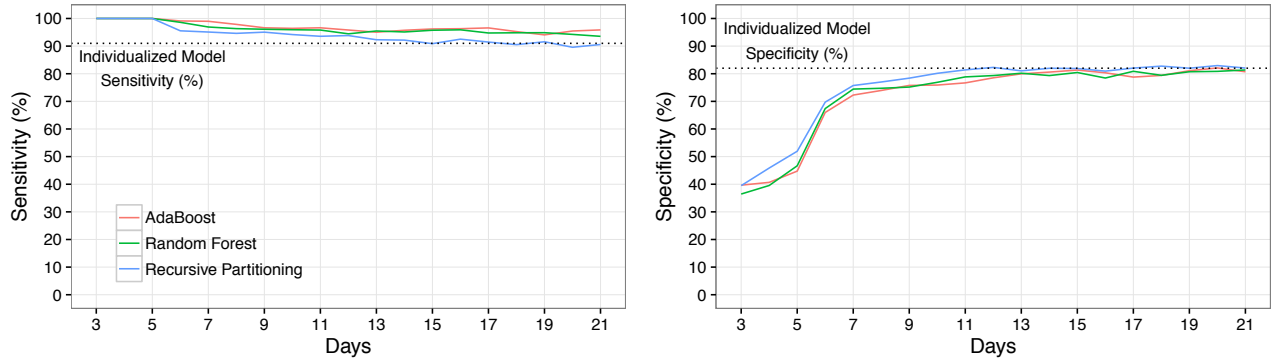

Figure 6: Prediction results for the online learning approach.

specificity of the generic models with new features increase by only 5% and 3% respectively.

7 ONLINE LEARNING APPROACH

In the previous section we have discussed and evaluated the prediction models by using a batch learning method based on "static" data, which is collected, stored and only then processed. However, this approach has two key drawbacks when used on mobile phones *in-the-wild*. Firstly, in a real world scenario the training data is not initially available; it rather becomes available gradually as new notifications arrive. Therefore, an individualized model cannot be constructed until a sufficient amount of data is available. This is known as the *bootstrapping problem* [12]. Another limitation is that the model cannot adapt when users change their behavioral patterns and, thus, the prediction accuracy might be dramatically reduced.

To overcome these issues it is possible to adopt an online learning approach in which a model is periodically trained with the available data that is collected gradually. Although this approach does not solve the issue of initial bootstrapping completely, it still enables relatively fast learning, adaptation to the potential changes in the user's behavior and improvement of the performance over time (i.e., the average prediction accuracy can be enhanced gradually as more and more training data becomes available). We construct individualized models with the same three algorithms: AdaBoost, Random Forest and Recursive Partitioning. We iteratively train the models with all the notifications collected by the end of each day and evaluate these models by using the notifications of the following day. More formally, on day D a model is built by using notifications from day 1 to day $D-1$ and it is evaluated by using the notifications from day D.

Figure 6 presents the prediction accuracy of the models with increasing number of days. The best achievable performance of the individualized prediction model trained with the batch learning approach is 82% specificity and 91% sensitivity (as discussed in the previous section). The results demonstrate that in just seven days all models become stable achieving a specificity of more than 70% and sensitivity of more than 95%.

8 DISCUSSION AND LIMITATIONS

According to the results, in our dataset around 44.8% of notifications are triggered by cross-platform applications and 28.59% of these notifications are handled on an alternative device (i.e., not handled on the mobile phone). However, this is not consistent for all users.

Instead, some users interact with notifications solely on their mobile phones, whereas others tend to use multiple devices for handling notifications.

The results also demonstrate that users' decisions in terms of handling notifications on a mobile or an alternative device are associated with their physical activity, location, application category, network connectivity and the device on which they handled the preceding notification. These features can be used to model users' behavior in terms of their interaction with notifications in multi-device environments. However, we have shown that it is more effective to train such models on individual data rather than on the entire set of users' data.

The main limitation of this work resides in the fact that we do not have data for each device accessed by a user but only for two classes of devices: personal phones and alternative devices (i.e., any device other than the phone). At the same time, the proposed model is highly generalizable to the case of multiple classes of devices. We would also like to point out that, from a practical point of view, it might be very difficult to deploy a real-world system that is able to capture *all* the interactions of a user with *all* the devices they access. Indeed, it might not be feasible to install notification loggers for privacy, legal or technical reasons (e.g., on workstations in companies, etc.). Moreover, we believe that this would be possible only for a subset of devices (or, more precisely, operating systems) for which a specific version of the application is available.

We would also like to underline the fact that some applications might be affected by a delay with respect to syncing cross-platform notifications (i.e., a delay between the user's handling of a notification in one of the devices and its automatic removal in all the other ones). The overall impact is probably negligible, but when this happens, we might fail to account for multiple interactions with duplicate notifications. Indeed, in the current study, it was not possible to identify such duplicate interactions as we do not have a notification interaction logger on all devices accessed by users.

9 CONCLUSIONS

In this paper we have presented a novel solution for intelligent notification delivery in multi-device environments by analyzing previous notification response behavior and a series of features that describe the user's context. In particular, we have shown that there is a significant relationship between user's behavior in terms of handling notifications and several contextual dimensions, including activity, location, network connectivity, application category and the device used for attending to the previous notification.

We have constructed and evaluated a set of prediction models, considering both generalized and personalized training approaches. We have shown that an individualized model is characterized by better prediction performance (i.e., 82% specificity and 91% sensitivity). We have also discussed the implementation of an online predictor that achieves a specificity of 70% and a sensitivity of 95% in just seven days.

The proposed mechanism could be considered as a building block for intelligent notification delivery in multi-device environments. The mechanism can be implemented as a server component that should be aware of all devices of each user that have subscripted to a given cross-platform application.

ACKNOWLEDGEMENTS

This work was supported through the EPSRC grants EP/L018829/2 and EP/P016278/1 at UCL and by The Alan Turing Institute under the EPSRC grant EP/N510129/1.

REFERENCES

[1] 2016. Android's NetworkInfo API. https://developer.android.com/reference/android/net/NetworkInfo.html.
[2] 2016. Android's Notification Listener Service. http://developer.android.com/reference/android/service/notification/NotificationListenerService.html.
[3] 2016. Google's Activity Recognition Application. http://developer.android.com/training/location/activity-recognition.html.
[4] 2018. Scikit-learn Package. https://scikit-learn.org.
[5] Piotr D Adamczyk and Brian P Bailey. 2004. If not now, when?: the effects of interruption at different moments within task execution. In *CHI'04*.
[6] Brian P Bailey and Joseph A Konstan. 2006. On the need for attention-aware systems: Measuring effects of interruption on task performance, error rate, and affective state. *Computers in Human Behavior* 22, 4 (2006), 685–708.
[7] Brian P Bailey, Joseph A Konstan, and John V Carlis. 2000. Measuring the effects of interruptions on task performance in the user interface. In *SMC'00*.
[8] Nicholas J Belkin and W Bruce Croft. 1992. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM* 35, 12 (1992), 29–38.
[9] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32.
[10] Edward Cutrell, Mary Czerwinski, and Eric Horvitz. 2001. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. In *Interact'01*.
[11] Mary Czerwinski, Edward Cutrell, and Eric Horvitz. 2000. Instant messaging and interruption: Influence of task type on performance. In *OZCHI'00*.
[12] Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.
[13] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. 2010. Diversity in smartphone usage. In *MobiSys'10*.
[14] Adrienne Porter Felt, Serge Egelman, and David Wagner. 2012. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *SPSM'12*.
[15] Joel E Fischer, Chris Greenhalgh, and Steve Benford. 2011. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *MobileHCI'11*.
[16] Joel E Fischer, Nick Yee, Victoria Bellotti, Nathan Good, Steve Benford, and Chris Greenhalgh. 2010. Effects of content and time of delivery on receptivity to mobile interruptions. In *MobileHCI'10*.
[17] Jerome H Friedman. 1976. A recursive partitioning decision rule for nonparametric classification. *IEEE Trans. Comput.* 26, 4 (1976), 404–408.
[18] Sukeshini A Grandhi and Quentin Jones. 2009. Conceptualizing interpersonal interruption management: A theoretical framework and research program. In *HICSS'09*.
[19] Joyce Ho and Stephen S. Intille. 2005. Using Context-Aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices. In *CHI'05*.
[20] Eric Horvitz, Paul Koch, and Johnson Apacible. 2004. BusyBody: creating and fielding personalized models of the cost of interruption. In *CSCW'04*.
[21] Jimmy Lin, Salman Mohammed, Royal Sequiera, and Luchen Tan. 2018. Update Delivery Mechanisms for Prospective Information Needs: An Analysis of Attention in Mobile Users. In *SIGIR'18*.
[22] Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. 2016. PrefMiner: Mining User's Preferences for Intelligent Mobile Notification Management. In *UbiComp'16*.
[23] Abhinav Mehrotra, Sandrine R. Müller, Gabriella M. Harari, Samuel D. Gosling, Cecilia Mascolo, Mirco Musolesi, and Peter J. Rentfrow. 2017. Understanding the role of places and activities on mobile phone interaction and usage patterns. *IMWUT* 1, 3 (2017).
[24] Abhinav Mehrotra and Mirco Musolesi. 2017. Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges. *arXiv preprint:1711.10171* (2017).
[25] Abhinav Mehrotra, Mirco Musolesi, Robert Hendley, and Veljko Pejovic. 2015. Designing Content-driven Intelligent Notification Mechanisms for Mobile Applications. In *UbiComp'15*.
[26] Abhinav Mehrotra, Veljko Pejovic, Jo Vermeulen, Robert Hendley, and Mirco Musolesi. 2016. My Phone and Me: Understanding People's Receptivity to Mobile Notifications. In *CHI'16*.
[27] Abhinav Mehrotra, Fani Tsapeli, Robert Hendley, and Mirco Musolesi. 2017. MyTraces: Investigating Correlation and Causation between Users' Emotional States and Mobile Phone Interaction. *IMWUT* 1, 3 (2017).
[28] Karl Pearson. 1900. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can

be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 50, 302 (1900), 157–175.

[29] Veljko Pejovic and Mirco Musolesi. 2014. InterruptMe: designing intelligent prompting mechanisms for pervasive applications. In *UbiComp'14*.

[30] Martin Pielot, Bruno Cardoso, Kleomenis Katevas, Joan Serrà, Aleksandar Matic, and Nuria Oliver. 2017. Beyond interruptibility: Predicting opportune moments to engage mobile phone users. *IMWUT* 1, 3 (2017), 91.

[31] Martin Pielot, Karen Church, and Rodrigo de Oliveira. 2014. An in-situ study of mobile phone notifications. In *MobileHCI'14*.

[32] Martin Pielot, Rodrigo de Oliveira, Haewoon Kwak, and Nuria Oliver. 2014. Didn't you see my message?: predicting attentiveness to mobile instant messages. In *CHI'14*.

[33] Laura Elena Raileanu and Kilian Stoffel. 2004. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence* 41, 1 (2004), 77–93.

[34] Gunnar Rätsch, Takashi Onoda, and K-R Müller. 2001. Soft margins for AdaBoost. *Machine Learning* 42, 3 (2001), 287–320.

[35] Adam Roegiest, Luchen Tan, and Jimmy Lin. 2017. Online in-situ interleaved evaluation of real-time push notification systems. In *SIGIR'17*.

[36] Adam Roegiest, Luchen Tan, Jimmy Lin, and Charles LA Clarke. 2016. A platform for streaming push notifications to mobile assessors. In *SIGIR'16*.

[37] Stephanie Rosenthal, Anind K Dey, and Manuela Veloso. 2011. Using decision-theoretic experience sampling to build personalized mobile phone interruption models. In *PerCom'11*.

[38] Alireza Sahami Shirazi, Niels Henze, Tilman Dingler, Martin Pielot, Dominik Weber, and Albrecht Schmidt. 2014. Large-scale assessment of mobile notifications. In *CHI'14*.

[39] Alireza Sahami Shirazi and Niels Henze. 2015. Assessment of Notifications on Smartwatches. In *MobileHCI'15 Adjunct*.

[40] Luchen Tan, Adam Roegiest, Jimmy Lin, and Charles LA Clarke. 2016. An exploration of evaluation metrics for mobile push notifications. In *SIGIR'16*.

[41] Fani Tsapeli and Mirco Musolesi. 2015. Investigating causality in human behavior from smartphone sensor data: a quasi-experimental approach. *EPJ Data Science* 4, 1 (2015), 1–15.

[42] Dominik Weber, Alexandra Voit, Philipp Kratzer, and Niels Henze. 2016. In-situ investigation of notifications in multi-device environments. In *UbiComp'16*.

[43] Dominik Weber, Alexandra Voit, Huy Viet Le, and Niels Henze. 2016. Notification dashboard: enabling reflection on mobile notifications. In *MobileHCI'16 Adjunct*.