Interpretable Machine Learning for Mobile Notification Management: An Overview of PrefMiner

Abhinav Mehrotra, University College London and The Alan Turing Institute, UK

Robert Hendley, University of Birmingham, UK

Mirco Musolesi, University College London and The Alan Turing Institute, UK

Mobile notifications are increasingly used by a variety of applications to inform users about events, news or just to send alerts and reminders to them. However, many notifications are neither useful nor relevant to users' interests and, also for this reason, they are considered disruptive and potentially annoying. PrefMiner is a novel interruptibility management solution that learns users' preferences for receiving notifications based on automatic extraction of rules by mining their interaction with mobile phones. PrefMiner aims at being intelligible and interpretable for users, i.e., not just a "black-box" solution, by suggesting rules to users who might decide to accept or discard them at run-time. The design of PrefMiner is based on a large scale mobile notification dataset and its effectiveness is evaluated by means of an in-the-wild deployment.

Today's mobile phones are highly personal devices characterized by always-on connectivity and high-speed data processing. These affordances make it a unique platform for applications harnessing the opportunity of real-time information delivery. A variety of applications are available on the app stores that enable users to subscribe to numerous information channels and actively receive information through notifications.

Past studies have shown that users are willing to tolerate some interruptions from notifications, so that they do not miss any important information [5]. However, their willingness is, in a sense, exploited by mobile applications as these trigger a plethora of notifications continuously [6]. Given the potentially large number of notifications, users do not accept all of them as their receptivity relies on the content type and the sender of the messages [6, 7]. Some examples of such notifications are promotional emails, game invites on social networks and predictive suggestions by applications. Users mostly dismiss (i.e., swipe away without clicking) notifications that are not useful or relevant to their interests [4]. On receiving such irrelevant or unwanted notifications users get annoyed. This could result in uninstalling the corresponding application [3].

Most of the previous studies propose interruptibility management systems that leverage the concept of anticipatory computing [8] to predict opportune moments

by using context [9] and content [6]. However, in order to reduce the level of disruption, an interruptibility management system should not just try to deliver notifications at opportune moments but also stop notifications that are not useful, or are uninteresting or irrelevant for the user.

Starting from this consideration, we designed and implemented *PrefMiner* – an intelligent interruptibility management solution that learns the types of information users prefer to receive via notifications in different situations. Moreover, in order to design a system that is *intelligible*, we implement a mechanism for mining association rules [1] and making the discovered rules transparent to users so that they can check their appropriateness. We believe that PrefMiner represents one of the first attempts in building an intelligent mobile system based on interpretable machine learning.

Mining Users' Preferences

In a classic study [2], Clark suggested that users' negative response to an interruption can be of two types: (a) acknowledge it and agree to handle it later; (b) decline it (explicitly refusing to handle it). PrefMiner is based on the second type of reaction: the system learns the different types of interruptions that users explicitly refuse by dismissing notifications. In this way PrefMiner can identify the notifications that are not useful for the users in specific situations and stop the operating system from triggering alerts related to them.

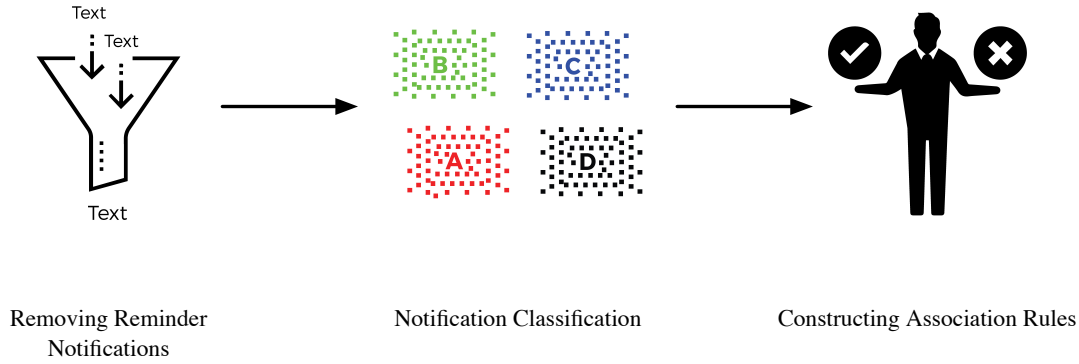

Figure 1. The process of learning user's preferences.

As shown in Figure 1, the process of mining user preferences consists of three steps discussed as follows.

Removing Reminder Notifications

The first step consists of identifying a particular class of notifications that are always dismissed but they should be shown in any case to users. As discussed earlier, notifications are dismissed if they are not found to be useful or relevant to the user's interest [4]. However, some notifications are dismissed because they do

not require any further action from the user. These notifications should not be automatically filtered, since they might be relevant for users, even if they are always dismissed. We refer to such notifications as *reminder notifications*. Alarm, calendar event and battery status notifications are some common examples of reminder notifications.

Notification Classification

In the second step we categorize each notification based on the information contained in it. Since a user might be interested to receive some but not all types of notification triggered by a specific application, PrefMiner performs clustering by considering their titles. A notification title is a short sentence that gives a glimpse of the information contained in it.

Constructing Association Rules

In order to discover rules about the user's preferences for receiving notifications, we use the AIS algorithm [1] – a method for mining data to discover statistical relationships between variables. An association rule is represented as $X \rightarrow Y$, where X is defined as the antecedent and Y as the consequent. To better understand the concept of association rules let us consider an example where the user: (i) always dismisses Twitter notifications for who to follow; (ii) accepts Facebook birthday reminder notifications only in the morning while she is at home; (iii) does not accept WhatsApp notifications from Alice while at work. Assuming that notifications about the Twitter suggestion, Facebook birthday reminder and WhatsApp from Alice are classified in the classes N_1, N_2 and N_3 respectively, the following association rules would represent the user's preferences in this case:
$\{N_1\} \rightarrow \{Dismiss\}$
$\{N_2, Home, Morning\} \rightarrow \{Accept\}$
$\{N_2, Home, Afternoon\} \rightarrow \{Dismiss\}$
$\{N_2, Home, Evening\} \rightarrow \{Dismiss\}$
$\{N_2, Home, Night\} \rightarrow \{Dismiss\}$
$\{N_2, Work\} \rightarrow \{Dismiss\}$
$\{N_2, Other\} \rightarrow \{Dismiss\}$
$\{N_3, Home\} \rightarrow \{Accept\}$
$\{N_3, Other\} \rightarrow \{Accept\}$
$\{N_3, Work\} \rightarrow \{Dismiss\}$

The rules are extracted by calculating the ratio between the number of times X and Y co-occur and the number of data-instances present in the given data (usually referred to as *support*) and the ratio between the number of times Y co-occurs with X and the number of times X occurs in the given data (usually referred to as *confidence*) [1].

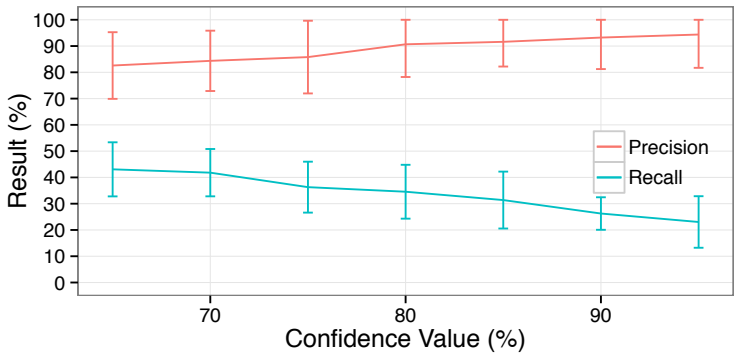

Figure 2. Performance of PrefMiner for predicting user's receptivity to notifications.

Evaluation of the Rule-based Mechanism

Before assessing the proposed mechanism in a real-world scenario, we first evaluate it on an existing dataset that we collected during the My Phone and Me study (to investigate the affect of ongoing task's characteristics on the perceived interruption) [7].

For mining association rules about the user's preferences we rely on the following features: notification response (i.e., the user's response to a notification), notification type, arrival time, activity, and location of the user when the notification arrived. We constructed association rules by using different combinations of these features. The consequent of these rules is restricted to contain only the notification response and the antecedent is restricted to never contain the notification response. We introduce this constraint because we are only interested to predict the acceptance of a notification, therefore other items in the consequent would be of no use and just add extra computational load.

We evaluated the discovered rules by using a 10-fold cross validation approach for different values of *confidence* level. However, the minimum *support* level is set to ensure that the notification interaction pattern covered in each rule has occurred at least once in two days. In order to assess the discovered association rules, we compared the predicted response with the actual response (i.e., the ground truth). The results show that increasing the confidence of association rules decreases the recall but improves the precision. This implies that by increasing the confidence fewer but more reliable rules are discovered. Moreover, we found that the association rules that are constructed by using the notification response, type and location perform better than rules constructed with other combinations. As shown in Figure 2, its recall goes up to around 43% without dropping the precision below 79%. Even by combining all features together, the performance of rules do not improve. Consequently, our results provide evidence that the user's preference for receiving notifications does not depend on the activity and arrival time, but on the type of information it contains and the location of the user.

Optimizing the System for High Precision

The key requirement for deploying an interruptibility management mechanism *in-the-wild* is that it should never stop/defer useful notifications. Therefore, while designing it we should aim to have fewer false-negatives (i.e., incorrectly predicting a notification as non-interesting for the user) which could be achieved by ensuring that the precision remains close to 100%. At the same time, the interruptibility management mechanism should also achieve a significant value for recall in order to demonstrate its efficacy in filtering notifications that are not useful or relevant to the user's interest. We could not obtain a high recall value because not all dismissed notifications are non-useful. Instead, some notifications are dismissed because they do not require any further actions, such as the final message of a chat conversation (e.g., *"Ok, bye!"*).

In general, this is a fundamental trade-off in the design of this class of systems. In our deployment, which we will describe later in this article, we set the parameters of the rule extraction algorithm in order to obtain a 35% recall with a precision of more than 90% at a confidence of 80%.

MyPref Library

We funnel our findings into the MyPref library that is implemented for the Android OS and released as an open source project [1]. The goal is to provide developers with a practical generic tool for intelligent rule-based notifications that can be integrated in any application, hiding at the same time the complexity related to the prediction mechanisms. The MyPref library abstracts the functionalities of the proposed interruptibility mechanism through a set of intuitive API primitives.

Moreover, to enable an overlying application to facilitate the transparency of the prediction mechanism to the users, the library makes the rules human understandable by replacing each notification type with the most frequent words of the relevant notification cluster (we refer to these words as *keywords*). For instance, the keywords from the cluster of Facebook's birthday reminder notifications (such as *"Today is Alice's birthday."* and *"Alice and Chris have birthdays today. Help them have a great day!"*) would be *"today"* and *"birthday"*.

[1]https://github.com/AbhinavMehrotra/PrefMiner

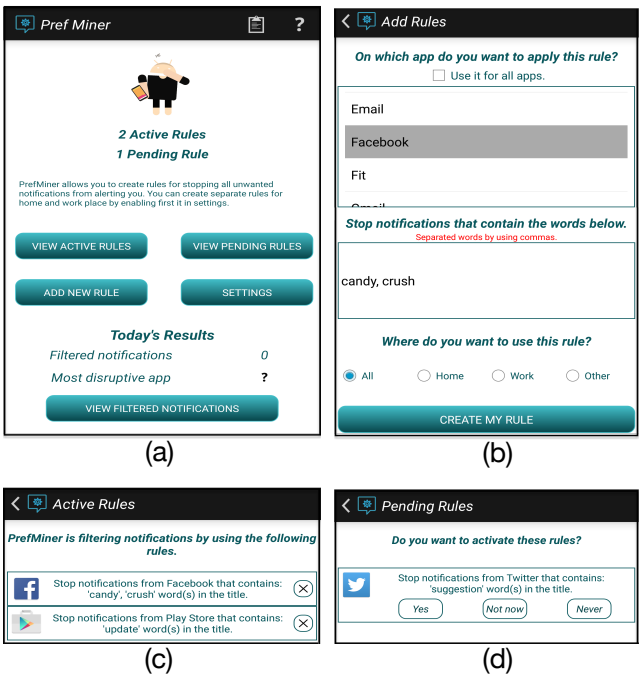

Figure 3. PrefMiner application: (a) main screen, (b) self-rule creation, (c) active rules, (d) pending rules.

In-the-wild Evaluation

We implemented the PrefMiner mobile application using MyPref (see Figure 3) and released it through Google Play Store in order to conduct an in-the-wild evaluation. The application continuously collects the notification data and binds the user's current location to each data instance. The rules are constructed every day when the phone is in charging mode and not in use so that the application does not directly affect users' mobile experience.

As shown in Figure 3.a the newly discovered rules are presented to the users in a human-readable format to get their approval before adopting them. To convert a rule into a human-readable format, we use the application name, the notification cluster identifier (i.e., the keywords provided by the library as a replacement for the notification type) and location. An example of such rules is the following: *"Stop notifications from Facebook that contain 'candy' and 'crush' words in the title"*.

During the 15-day study, PrefMiner suggested 179 rules to the participants out of which 102 rules (i.e., 56.98%) were accepted. Overall, around 70% of the users accepted 50% (and above) of the suggested rules. The results also show that the average number of notifications that are successfully filtered everyday is 12 (with the standard deviation equal to 8) and, thus, minimizes the perceived disruption for handling irrelevant notifications.

Conclusions

To the best of our knowledge, PrefMiner represents the first intelligent mobile notification management solution with the goal of making the underlying mechanisms intelligible for the users. More in general, it also represents one of the first examples of interpretable machine learning applied to mobile systems design. Indeed, most of the existing work in our community is based on machine learning mechanisms that are essentially implemented as "black-boxes". We believe that in order to design intelligent systems that are usable and acceptable for users, making the algorithmic decisions understandable is of paramount importance. We hope that our experience in designing PrefMiner will be useful for other researchers and practitioners for the design of the next-generation mobile systems.

[1] Agrawal, R. et al. 1993. Mining association rules between sets of items in large databases. *SIGMOD'93* (June 1993).

[2] Clark, H.H. 1996. *Using language*. Cambridge University Press.

[3] Felt, A.P. et al. 2012. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. *SPSM'12* (October 2012).

[4] Fischer, J.E. et al. 2010. Effects of content and time of delivery on receptivity to mobile interruptions. *MobileHCI'10* (September 2010).

[5] Iqbal, S.T. and Horvitz, E. 2010. Notifications and awareness: A field study of alert usage and preferences. *CSCW'10* (February 2010).

[6] Mehrotra, A. et al. 2015. Designing Content-driven Intelligent Notification Mechanisms for Mobile Applications. *UbiComp'15* (September 2015).

[7] Mehrotra, A. et al. 2016. My Phone and Me: Understanding People's Receptivity to Mobile Notifications. *CHI'16* (April 2016).

[8] Pejovic, V. and Musolesi, M. 2015. Anticipatory mobile computing: A survey of the state of the art and research challenges. *ACM Computing Surveys*. 47, 3 (2015), 1–47.

[9] Pejovic, V. and Musolesi, M. 2014. InterruptMe: Designing intelligent prompting mechanisms for pervasive applications. *UbiComp'14* (September 2014).