A FRAMEWORK FOR INTELLIGENT MOBILE NOTIFICATIONS

by

ABHINAV MEHROTRA

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
College of Engineering and Physical Sciences
The University of Birmingham
January 2017

Abstract

Mobile notifications provide a unique mechanism for real-time information delivery systems to users in order to increase its effectiveness. However, real-time notification delivery to users via mobile phones does not always translate into users' awareness about the delivered information because these alerts might arrive at inappropriate times and situations. Moreover, notifications that demand users' attention at inopportune moments are more likely to have adverse effects and become a cause of potential disruption rather than proving beneficial to users. In order to address these challenges it is of paramount importance to devise intelligent notification mechanisms that monitor and learn users' behaviour for maximising their receptivity to the delivered information and adapt accordingly.

The central goal of this dissertation is to build a framework for intelligent notifications that relies on the awareness of users' context and preferences. More specifically, we firstly investigate the impact of physical and cognitive (including emotional states and task engagement) contextual features on users' attentiveness and receptivity to notifications. Secondly, we construct and evaluate a series of models for predicting opportune moments to deliver notifications and mining users' notification delivery preferences in different situations. Finally, we design and evaluate a model for anticipating the right device notifications in cross-platform environments.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisors Mirco Musolesi and Robert Hendley for their support encouragement, and guidance throughout my PhD. Their guidance during these years has been invaluable. They have helped me to explore new ideas, critically evaluate my work, and express my work clearly in the form publications. Their enthusiastic encouragement and support helped me a lot in my personal and professional development. Mirco has introduced me to research during my MSc project which motivated me to do a PhD, thank you very much for that.

I would like to acknowledge my gratitude to the members of our research group at the University of Birmingham and University College London who contributed to this thesis by collaborating, helping me to conduct my research work and providing feedback on my work. I thank you all (Antonio Lima, Beatrice Perez, Christoph Stich, Fani Tsapeli, Giovanni Quattrone, Luca Canzian, Luca Rossi, Matthew Williams, and Veljko Pejovic) for many lively and interesting conversations. Special thanks to Veljko with whom I have collaborated from the very outset of my PhD. The learning I had from this collaboration is priceless. His comments and suggestions have informed my research in numerous ways. Also, I thank all the friends I have met in these years. In particular, thanks to Gurchetan for his great support and guidance. I thank my research monitoring group members, Rami Bahsoon and Russell Beale for their support, comments and suggestions.

I would like to express my deepest appreciation to my parents, wife and my whole family for loving me and supporting me to achieve my dreams and ambitions. Especial thanks to my brother Nitin, who always encouraged me to pursue my interests. Also,

a big thanks to my wife Ayushi, who has been a source of constant encouragement and support during the highs and lows of my PhD journey. I dedicate this thesis to you all.

STATEMENT OF COLLABORATION

The second part of the Chapter 3 is my work in collaboration with another PhD student (Fani Tsapeli). Most of the work (including experimental design, application development, data collection, data preprocessing and correlation analysis) was done by me. However, Fani Tsapeli provided support to perform the causality analysis by using her framework presented in [TM15].

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Overview

Mobile phones today have become the most ubiquitous personal computing devices on the planet [Kru16]. Cellular network coverage has reached 96.8% of the world population and this number even reaches 100% of the population in developed countries [Int15]. These devices have transformed over a period of time from merely communication tools to smart and highly personal devices that are able to assist us in a variety of day-to-day situations.

Besides being an indispensable part of our daily life and pervasive in nature, mobile phones also have the capability to be always connected to the Internet. These features of the mobile phone have made it a unique platform for fetching or receiving information at anytime and anywhere. Moreover, such an opportunity has expanded the potential of mobile applications to deliver information about a variety of events ranging from emails to updates on online social networks and from advertisements about a product to interventions for behaviour change programmes.

In order to ensure the real-time awareness of users about the delivered information, mobile operating systems rely on *notification mechanisms* that steer users' attention towards the delivered information through audio, visual and haptic signals. Notifications are the cornerstone of push-based information delivery via mobile phones as they allow applications to harness the opportunity of steering users' attention in order to increase

the impact of the delivered information. Notifications are presented in a unified fashion by almost all mobile operating systems. Usually, notifications from all applications are listed in a notification-bar located at the top of the phone's screen. In order to provide a quick glimpse of the delivered information to the users, they present a brief summary including the sender, information highlights and time of delivery.

Mobile notifications are generated by humans as well as machines. The former are generated by recipient's social connections generally through chat and email applications for instantiating communication between two or more persons. Whereas, in the case of the latter, messages are generated in an automatic fashion by the system processes or the native applications, such as system monitoring utilities, scheduled reminders and promotional advertisements.

Furthermore, since today's mobile phones come equipped with a plethora of sophisticated sensors, the data generated therefrom is exploited by applications for not just improving usability but also to capture users' context. Here, the user's context is defined as *"any information that can be used to characterise the situation of an entity"* [Dey01]. Contextual information can be used by sophisticated applications for proactively signalling users about the occurance of events that are associated with their context. Some common example of context-based notifications are collocation-based advertisements [AGKO04, FS10] and context-based suggestions [LCC+11, WCC+12].

1.2 Problem Definition and Objectives

An increasing number of smartphone applications actively push information to the users. People receive numerous notifications arriving autonomously at anytime during the day through their mobile phones [SSHD+14, MMHP15]. Such proactive services are indeed beneficial to the users, facilitating task switching and keeping users aware of a number of information channels in an effortless manner. However, at the same time, due to the pervasive nature of mobile phones, these notifications often arrive at inconvenient

moments without any knowledge about recipients' situation and their willingness to get interrupted.

Psychological studies have found that wrongly timed notifications come with a cost of interruption to the on-going task [MN86, ZRLK99]. Such an interruption can adversely affect task completion time [CCH01, CCH00a, MBDT02], error rate [BKC00] and the affective state of the user [AB04, BK06]. These experimental studies were conducted in a controlled environment to investigate the effects of generic interruptions. This suggests that these findings may be generalisable to desktop and mobile interruptions. Moreover, some recent studies have demonstrated the negative affect of mobile interruptions of the affective state and productivity of the user [KPD16, PCdO14]. At the same time, recent studies have shown that this problem is exacerbated by the fact that cross-platform applications prompt the users on multiple devices at the same time [WVKH16]. This fact makes these notifications potentially even more annoying.

On the other hand, previous studies have shown that users are willing to tolerate some interruptions from notifications, so that they do not miss any important information [IH10]. However, their willingness is, in a sense, exploited by mobile applications as these trigger a plethora of notifications continuously [MMHP15]. 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 [MMHP15, MPV+16]. Users mostly dismiss (i.e., swipe away without clicking) notifications that are not useful or relevant to their interests [FYB+10, SSHD+14]. Some examples of such notifications are promotional emails, game invites on social networks and predictive suggestions by applications. Furthermore, studies have shown that users get annoyed by receiving such irrelevant or unwanted notifications which could result in uninstalling the corresponding application [FEW12, SSHD+14]. The above findings provide evidence that, in order to reduce the level of disruption, we need to build smart notification mechanisms for alerting users and sending information at appropriate moments.

At the same time, sensors embedded in mobile phones allow us to capture contextual information that can be used to both monitor and learn users' behaviour. Indeed, the interaction of users with notifications is extremely complex and depends on various contextual aspects, some of which can be captured through mobile sensors. Consequently, the knowledge about the recipient's notification-interaction behaviour and corresponding context offer a significant opportunity to tackle the problem of designing a framework for intelligent notifications, which can: (i) understand the factors associated with users' receptivity to notifications and infer opportune moments for notification delivery; (ii) identify and hide or remove notifications that are not useful, or are uninteresting or irrelevant for the user; and (iii) forecast the best medium to deliver cross-platform notifications.

1.3 Motivation

The key motivation of this work is to contribute to the design of calm mobile phone technology, which aims to reduce the "excitement" of information overload by providing users with the information that is in their interest and only when it is needed. Mark Weiser and John Seely Brown describe calm technology as "a technology which informs the users but does not demand their focus or attention" [WB97]. They envisioned that calm technology will not only relax the user by hiding irrelevant information but also allow more information to exist there and ready for users when needed. In order to avoid making the existing mobile phone technology as intrusive, we need to design mobile notifications that are aware of users' context, their willingness to be interrupted and the right medium to initiate the interruption.

Furthermore, the resulting intelligent notification mechanisms should not only enable us to achieve unobtrusive mobile technology, but also facilitate effective information delivery. Overall, this would offer a great opportunity to researchers and practitioners for influencing human behaviour through mobile phone-based behaviour change interventions [LPR+13, PM14a]. For instance, monitoring and influencing obesity, stress and

depression are some of the health and well-being challenges that can be successfully tackled by triggering timely and relevant interventions. Additionally, intelligent notification systems could also unlock the mobile phone's potential to be a cornerstone for building innovative applications in areas spanning from commerce [FS10] to public safety [WCC+12].

1.4 Thesis and Research Questions

As discussed in the previous sections, although mobile notifications[1] are extremely beneficial to the users, they are a cause of potential disruption as they often require users' attention at inopportune moments. In order to realise unobtrusive technology in the case of mobile notifications, we must make them aware of the recipient's situation.

Consequently, the primary thesis of this dissertation is that the analysis of human traces obtained through mobile phone sensing can be used to build mechanisms for capturing the human-notification interaction behaviour and embedding intelligence in notification mechanisms.

In order to support the above statement, this dissertation presents a series of empirical studies to design specific components of the proposed framework. More specifically, this dissertation investigates the following research questions:

- **Research Question 1:** What are the physical and cognitive contextual features that make people attentive and receptive to mobile notifications?
- **Research Question 2:** Can we predict the opportune moments to deliver notifications given a user's context?

[1]Details about Android notifications are discussed in the appendix of this thesis.

- **Research Question 3:** Can we infer users' preferences to receive notifications in different situations?

- **Research Question 4:** On which medium should a notification be delivered in multi-device environments?

1.5 Contributions and Thesis Outline

This dissertation aims to study and model user behaviour through a data-driven approach. More specifically, we investigate the use of mobile phone sensing to build mechanisms for intelligent notifications. This dissertation is organised as follows. In Chapter 2, we review the previous studies in the area of mobile interruptibility management and demonstrate how this thesis advances the state of the art in mobile notification management. The details of these contributions are discussed in the following chapters.

The key contributions and corresponding mapping with the chapters can be summarised as follows:

- **Contribution 1: Understanding People's Attentiveness and Receptivity to Mobile Notifications.**

  In Chapter 3, we present two in-situ studies of mobile interruptibility focusing on the effects of cognitive context and notification design factors on users' perception, attentiveness and receptivity to mobile notifications. The overall contributions of this chapter are twofold. First, we have confirmed the validity of some past desktop interruptibility studies [CCH00b, CCH00a, CCH01] to show that ongoing task's complexity and completion level influences the perceived interruption in a mobile setting. Second, for the first time, we have investigated the role of notification presentation, sender-recipient relationship and emotional states for modelling interruptibility. Overall, the findings of this chapter are wide-ranging and they represent the basis for the design of the proposed framework for intelligent notifications.

- **Contribution 2: Designing A Context and Content-driven Mechanism for Predicting Interruptible Moments.**

  In Chapter 4, we present an in-situ study of mobile user interruptibility with the aim of investigating how users behave when they receive specific types of content through mobile notifications arriving in different contexts and from different senders. The study shows that not all the notifications are disruptive. Instead, it is the relevance of the interruption content in the recipient's current context that partly defines the disruptiveness of an interruption. We present the design and implementation of a machine learning model that uses both content and context together for predicting the most opportune moment for the delivery of *in-the-wild* notifications (i.e., the real-world notifications received by users) carrying specific types of information. Finally, we show that a machine learning approach leads to more accurate predictions of users' interruptibility than alternative ones based on user-derived interruptibility rules.

- **Contribution 3: Mining Users' Preferences for Receiving Notifications.**

  In Chapter 5, we present the design, implementation and evaluation of a novel interruptibility management solution that learns users' preferences for receiving notifications based on automatic extraction of rules by mining their past interactions. In simple words, the proposed solution learns the types of information users prefer to receive via notifications in different situations and filters out those that are not opportune in a given context. Another important contribution of this study is to design a "customisable system" rather than a "black-box" solution. We present the design and implementation of a mechanism for mining association rules and make the discovered rules transparent to users so that they can check their appropriateness. This allows the system to improve its effectiveness by discarding rules that are not correct, according to users.

- **Contribution 4: Predicting the Right Device on Which to Deliver Notifications in Cross-platform Environments.**

  As discussed above, we conducted studies to infer opportune moments and to learn the types of notifications users prefer to receive in different contexts. In Chapter 6, we first investigate the relationship between users' behaviour in terms of handling notifications in a multi-device environment and several contextual dimensions. Then, we present the design of individualised and generic models to predict the device on which the user will handle a notification in a given context. Finally, we show that user clustering techniques can be applied to build predictors for initial model bootstrapping.

In Chapters 7, we conclude this thesis with a brief summary of its key contributions in the area of interruptibility management. Finally, we present some proposals for future work.

1.6 Publications

The work presented in this dissertation has also been published or are in submission for publications in the following conference and workshop papers. We also published a book chapter on these topics.

1.6.1 Publications Related to this Dissertation

Conference Papers

- A. Mehrotra, R. Hendley and M. Musolesi, PrefMiner: Mining User's Preferences for Intelligent Mobile Notification Management. UbiComp'16. Heidelberg, Germany, September 2016. [**Best Paper Award**] [MHM16a]

- A. Mehrotra, V. Pejovic, J. Vermeulen, R. Hendley and M. Musolesi, My Phone and Me: Understanding People's Receptivity to Mobile Notifications. CHI'16. San Jose, CA, USA, April 2016. [MPV+16]

- A. Mehrotra, M. Musolesi, R. Hendley and V. Pejovic, Designing Content-driven Intelligent Notification Mechanisms for Mobile Applications. UbiComp'15. Osaka, Japan, September 2015. [MMHP15]

Workshop Papers

- A. Mehrotra, J. Vermeulen, V. Pejovic and M. Musolesi, Ask, But Don't Interrupt: The Case for Interruptibility-Aware Mobile Experience Sampling. UbiComp'15 Adjunct. Osaka, Japan, September 2015. [MVPM15]

- V. Pejovic, A. Mehrotra and M. Musolesi, Investigating The Role of Task Engagement in Mobile Interruptibility. MobileHCI'15 Adjunct. Copenhagen, Denmark, August 2015. [MHM16b]

- A. Mehrotra, J. Vermeulen, R. Hendley, and M. Musolesi. Challenges In Managing Multi-Device Notifications. In Cross-Surface'15, Madeira, Portugal, November 2015.

Under Submission

- A. Mehrotra, F, Tsapeli, R. Hendley and M. Musolesi, MyTraces: Investigating Correlation and Causation between User's Emotional State and Mobile Phone Interaction. Submitted to UbiComp'17. Maui, Hawaii, USA, September 2017.

- A. Mehrotra, R. Hendley and M. Musolesi, NotifyMeHere: Intelligent Notification Delivery in Multi-Device Environments. Submitted to UbiComp'17. Maui, Hawaii, USA, September 2017.

Book Chapters

- A. Mehrotra and M. Musolesi, Sensing and Modeling Human Behavior Using Social Media and Mobile Data. Book Chapter. (To appear)

- V. Pejovic, A. Mehrotra and M. Musolesi, Anticipatory Mobile Digital Health: Towards Personalised Proactive Therapies and Prevention Strategies. Book Chapter in Mihai Nadin (Ed.) Anticipation and Medicine. Springer. 2017.

CHAPTER 2

BACKGROUND

2.1 Interruptions

2.1.1 Definitions

The concept of *interruption* has been defined and interpreted in different ways by researchers working in different communities. For example, in linguistics, an interruption has been defined as:

"*A piece of discourse that breaks the flow of the preceding discourse. An interruption is in some way distinct from the rest of the preceding discourse; after the break for the interruption, the discourse returns to the interrupted piece of discourse [GS86].*"

In psychology, an interruption has been defined as:

"*An event that breaks the coherence of an ongoing task and blocks its further flow. However, people can resume the primary task that has been interrupted once the interruption is removed [ML02].*"

In computer science, an interruption has been defined as:

"*An event prompting transition and reallocation of attention focus from a task to the notification [MCSN03].*"

2.1.2 Types of Interruptions

Interruptions are pervasive in nature. As described by Miyata and Norman [MN86], in our everyday life we receive numerous interruptions that are both internal and external. In the following, we provide an overview of the definitions of these two types of interruptions as proposed by the authors of [MN86].

Internal Interruptions

An internal interruption occurs due to our own background thought process. More specifically, internal interruptions are the activities that people perform by breaking their focus of conscious attention.

External Interruptions

An external interruption is caused by the arrival of an event around us. Communication through computing devices or in-person is the fundamental source of external interruptions. External interruptions can be further divided into two categories based on the relevance of sources with the primary task:

- **Implicit Interruptions**: These are the interruptions that arrive from some process of a primary task. Such interruptions are mostly relevant to the current task, for example, an error message from the application which a person is interacting with.
- **Explicit Interruptions**: These are the ones that arrive from a process that does not belong to the ongoing task. Such an interruption causes an expected task switch from the ongoing task to a newly introduced task, for example, the arrival of a chat message while a person is interacting with a text editor application.

2.1.3 Our Definition of Interruptions

Previous theories about interruptions help us to gain a deeper insights about the meaning of interruption. We use this knowledge to derive a generic theoretical construct for *interruption*. Since the objective of this thesis is to reduce external interruption without considering internal interruptions. Therefore, we define interruption as *an unanticipated event that comes through a communication medium and has a potential to instigate a task switch and break the flow of the primary activity by capturing users' attention through a visual cue, auditory signal, or haptic alert.*

2.2 Sources of Interruptions

The aim of this work is to build a framework for intelligent mobile notifications. Therefore, the focus of this section is on the emergence of external interruptions and the ways in which people handle them.

2.2.1 Interruptions in Human-Human Discourse

In the human-human communication environment, when an interaction is initiated by a person (i.e., speaker), the listener usually provides feedback about the success or failure of the initiated communication [CS89]. Such feedback is merely a brief reaction through eye contacts, head nods or voice response. This acknowledgement informs the speaker whether the listener is welcoming and attending the communication or not.

As suggested by Clark and Schaefer [CS89], "*[f]or people to contribute to discourse, they must utter the right sentence at the right time*". The physical presence of a person enables the speaker to understand the right moment to initiate communication. However, the speaker does not usually have a reasonable understanding of the listener's cognitive situation. Consequently, people sometimes initiate communication at wrong moments, which often results in causing interruptions.

As suggested by Clark [Cla96], a listener can respond to such an interruption in four possible ways:

1. take-up with full compliance: responding to it immediately.
2. take-up with alteration: acknowledging and agreeing to handle it later.
3. decline: explicitly refusing to handle it.
4. withdraw: implicitly refusing to handle it by not providing an acknowledgement to the interruption.

On the other hand, once the conversation begins it does not always go error-free. As argued by Sacks et al. in [SSJ74], the turn-taking during the conversation is itself vulnerable to error. When a person speaks the other person listens, but sometimes unintentionally both start talking simultaneously and people try to coordinate their conversation in an appropriate way. However, this can be categorised as an implicit interruption (discussed earlier in this chapter) as there is no external source of interruption.

2.2.2 Interruptions in Human-Computer Interaction

Personal computing devices such as desktops, laptops and mobile phones, offer a great value to users by facilitating multifarious informative and computational functionalities salient to their daily requirements. However, the provision of a platform that runs multiple applications simultaneously, which are delivering various types of information from different channels, often leads to an environment that distracts users from their primary task[1]. This is due to the fact that the information delivered to the users is often not relevant to their primary task, which leads them to switch their attention from the application in focus towards the application being executed in the background that delivered the information. Moreover, applications leverage notifications in order to trigger alerts that

[1]Here a primary task can be any operation which the user is currently performing, which might or might not include interaction with a computing device.

try to gain users' attention towards the delivered information. Therefore, even though people try to ignore all interruptions and continue focusing on their primary task, they still receive cues about the newly delivered information, which might cause information overload [SVV99].

In 1997, by looking at the trend towards the development of "intelligent" computer system and technologies, McFarlane envisioned the hazards from such intelligent systems competing for users attention [McF97]. McFarlane, for the first time, investigated the strategies for counteracting interruptions caused by intelligent computer systems. He built a taxonomy based on theoretical constructs that are relevant to interruptions. This taxonomy identifies the following eight descriptive dimensions of human interruption:

(i) *source of interruption*: who triggered the interruption.

(ii) *individual characteristics of person receiving interruption*: receiver's perspective for getting interrupted.

(iii) *method of coordination*: technique used for determining when to interrupt based on the users' response.

(iv) *meaning of interruption*: what the interruption is about.

(v) *method of expression*: design aspect of the interruption.

(vi) *channel of conveyance*: medium of receiving the interruption.

(vii) *human activity changed by interruptions*: internal or external change in the recipient's conscious and physical activity.

(viii) *effect of interruption*: impact of interruption on an ongoing task and the user.

Instead of limiting the scope of his work to HCI, McFarlane built this taxonomy from an interdisciplinary perspective drawing upon the theories for human interruption discussed in the literature of many different domains. Each dimension of the taxonomy describes a unique aspect of human interruptibility.

In 1999, Latorella proposed an Interruption Management Stage Model (IMSM) that describes information processing stages on receiving interruptions [Lat99]. This model can be used to study information processing by humans and to identify the effects of interruptions in different stages of information processing. The model was designed with an assumption that recipients are engaged with an ongoing task with which they are familiar and that it can be resumed at any point. It comprises of three stages:

(i) *interruption detection*: when a user is engaged with the primary task, a salient alert is required in order to initiate an interruption.

(ii) *interruption interpretation*: on detection of an interruption, the user's attention is directed towards the interruption for further processing in order to interpret the requirements of the interrupting task.

(iii) *interruption integration*: in this final stage, the user integrates the interruption with the primary task by immediate or scheduled tasks switching.

In 2002, McFarlane and Latorella investigated users' behaviour on receiving interruption alerts (i.e., the first stage of IMSM model) [ML02]. They argued that user response to computer generated interruptions is similar to the response to interruptions during human dialogue as proposed by Clark [Cla96]. However, they suggested that Clark only considered the user response for detected interruptions, but that in the human-computer interaction setting these can also go undetected. Therefore, undetected interruptions are also additional aspects of user response, which should be considered when building interruption management system for computing environments.

McFarlane and Latorella proposed the following five key responses of users to an interruption arriving during the process of human-computer interaction:

(i) *oblivious dismissal*: the interruption is undetected and the interruption is not performed;

(ii) *unintentional dismissal*: the interruption is not interpreted and the interruption is not performed;

(iii) *intentional dismissal*: the interruption is interpreted, but the user decides not to perform the interrupting task;

(iv) *preemptive integration*: the interrupting task is initiated immediately, intruding upon the ongoing task, and is performed to completion before resuming the ongoing task;

(v) *intentional integration*: the interrupting task and the ongoing task are considered as a set, and the operator rationally determines how to integrate performance of the interrupting task.

2.3 Cost of Interruption

Interruptions are an inevitable part of our everyday life as it is hard to get through the entire day without being interrupted. As suggested by Zabelina et al. in [ZOP+15], people are sensitive to their surroundings and they receive more information through interruptions, which helps them to be creative. This represents a positive aspect of interruptions. At the same time, numerous studies [BEG98, CCH00b, BKC00, CCH01] have demonstrated that interruptions have a detrimental effect on users' memory, emotional and affective states, and ongoing task execution. In this section we discuss the cost associated with the arrival of interruptions at inopportune moments.

2.3.1 Impact on Memory

In 1927, Zeigarnik performed a classic psychological study [Zei27] (as cited in [Bad76]), which led to the definition of the *Zeigarnik Effect*, with the goal of examining the mechanisms of retrospective remembering with and without interruptions. In this study the participants were given a series of practical tasks, for instance sketching a vase and putting beads on a string. Some tasks were interrupted and others were carried out without any interruption. Participants were allowed to perform the task in any order and switch to

another task without completing the ongoing task, which could be taken up later. On the completion of all tasks, they were asked to do a recall test. The results of this study demonstrate that people can recall the content of interrupted tasks better compared to the content of uninterrupted tasks, which might indicate that people have selective memory associated to the interruptions they receive. This observed behaviour is usually referred to as the Zeigarnik Effect.

Although the Zeigarnik Effect suggests that interruptions are useful for retrospective memory, many other applied studies have argued that interruptions have an adverse impact on memory [DFAB93, GB89, BEG98, CCH01]. In particular, in [DFAB93] Dix et al. argued that humans can memorise only a limited list of tasks they have been engaged in due to the nature of their cognitive abilities. Moreover, if interrupted during a task, humans are likely to lose track of what they were doing. Following these suggestions, Edwards and Gronlund [BEG98] conducted an experiment to investigate the memory representation for the primary task after handling an interruption. Their study was orthogonal to the Zeigarnik Effect experiment as in the latter participants were not asked to resume or to remember where they left the primary task after the interruption. Through their experiment, Edwards and Gronlund demonstrated that people possess a stronger memory representation of an uninterrupted task as compared to an interrupted task. Moreover, they showed that people tend to take time in order to resume back to the primary task after an interruption.

2.3.2 Impact on Ongoing Task Performance

In 1981, Kreifeldt and McCarthy [KM81] argued that interruptions could have an adverse effect on the completion time and errors made while performing a computing task. They demonstrated that people perceive more disruption and become more prone to make errors on getting interrupted while performing a complex task compared to a simple task. Some years later, in 1989, Gillie and Broadbent [GB89] investigated, through a series

of experiments, the impact on the primary task by three aspects of an interruption: (i) length; (ii) similarity with the primary task; (iii) the action required to handle it. Their results show that people feel distracted when interruptions are similar to the primary task or if they are limited to complex tasks. However, the length of an interruption does not make it disruptive. However, Czerwinski et al. [CCH00b] found that people perceive less disruption if the interruption is highly relevant to the current task, which contradicts the findings of Gillie and Broadbent that people feel more distracted when interruptions are similar to the primary task [GB89].

In [BKC00] Bailey at al. investigated the impact of interruptions on the performance of an ongoing task. Their experiment utilised six types of web-based task: adding numbers, counting items, image comprehension, reading comprehension, registration and selection. Participants were interrupted when they were approximately in the midway to completion of each task. They were presented with a news summary or a stock-decision task as an interruption. Their findings show that: (i) people perform interrupted tasks slower as compared to non-interrupted tasks; (ii) perceived disruption varies as a function of the task type; and (iii) different types of interruption produce similar effects on task performance. Later, in another study [BKC01], Bailey's et al. demonstrated that the amount of disruption perceived also depends on the mental load of a user on the arrival of an interruption.

Czerwinski et al. in [CCH00a] investigated the impact of interruptions while performing different types of sub-tasks. Their results show that people perceive varying level of disruption while performing different sub-tasks. They proposed that deferring interruptions until a new subtask is detected could also reduce the perceived disruption. These findings were extended by Cutrell et al. [CCH01] to investigate the effects of instant messaging on different types of computing tasks. They found that the disruptiveness perceived is higher when users are engaged with tasks that require their attention.

2.3.3 Impact on Users' Emotional State

In [ZRLK99] Zijlstra et al. for the first time studied the effect of interruptions on users' psychological state. More specifically, they investigated whether interruptions produce an adverse effect on users' emotions and well-being, and raise their activeness level. They conducted a series of experiments by creating a simulated office environment for performing realistic text editing tasks. Their findings suggest that users' emotion and well-being are negatively impacted by interruptions, but they do not affect the activeness level.

In 2001, Bailey at al. investigated the effects of interruption on users' annoyance and anxiety levels [BKC01]. Through a series of experiments, they demonstrated that people experience annoyance on arrival of an interruption. The annoyance level experienced by users depends on the type of ongoing task, but not on the type of the interruption task. They also show that the increase in users' anxiety level is higher when they receive interruptions during a primary task as compared to the arrival of interruption on completion of the primary task.

In another study [AB04], Adamczyk and Bailey investigated the effects of interruptions arriving at particular moments during the execution of tasks in terms of users' emotional state and social attribution to the application of their primary task. In their experiment, participants were asked to perform tasks (such as text editing, searching and watching video) and a periodic news alert was triggered as an interruption. Their findings show that users experience annoyance and frustration on receiving interruptions. However, interruptions arriving at different moments have a varying impact on users' emotional state and social attribution to the primary task's application.

In a study about mobile notifications [KPD16], Kushlev et al. investigated whether interruptions coming from mobile phones cause lack of attention and hyperactivity symptoms associated with Attention Deficit Hyperactivity Disorder (ADHD). They asked participants to maximise interruptions by keeping notification alerts on and trying to mostly be within the reach of their phone. Later, participants were asked to minimise interruptions by keeping notification alerts off and their phones away. Their results show that

participants reported higher levels of inattention and hyperactivity during the first phase of the experiment. This suggests that by simply adjusting existing phone settings people can reduce inattention and hyperactivity levels.

2.3.4 Impact on User Experience

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. – Mark Weiser [Wei91]. The above quote from Mark Weiser's seminal paper [Wei91] provide a clear picture about his vision for ubiquitous computing. His goal was to design an environment with embedded unobtrusive computing and communication capabilities that can blend with users' day-to-day life. The two key aspects of his vision were: (i) effective use of the environment in order to fuse technology with it; (ii) making the technology disappear in the environment.

The second aspect of his vision focuses on the user experience and suggests making technology disappear from the user's consciousness. Another classic paper of Weiser [WB97] describes the disappearing technologies as *calm computing*. In this paper he suggested that the technology should enable the seamless provision of information to users without demanding their focus and attention. However, interruptions have a potential to drive technology away from Mark Weiser's vision because they not only create potential information overload but also demand user attention.

As suggested by Mark Weiser [Wei91], technology should be "transparent" to users so that they should not notice that they are interacting with computing devices. In one of the first works in this area, Kreifeldt and McCarthy compared the design of user interfaces in order to mitigate the effects of interruptions [KM81]. Their results suggest that the interaction design plays a role in affecting users to successfully resume the interrupted task. More specifically, their findings suggest that complex interfaces make it difficult to handle interruptions.

2.4 Individual Differences in Perceiving Interruptions

As defined earlier, an interruption is an unanticipated event that breaks the flow of execution of an ongoing task by demanding users to switch their attention to the interrupting task. This leads to a multitasking environment for users. Humans naturally have skills to handle interruptions and adapt to a multitasking environment; however, they do show individual differences in their ability to perform tasks in a dynamic multitasking environment [Atk53, Wei65, BW86, JH98].

In 1953, Atkinson investigated the role of people's motivation to recall the completed and interrupted tasks [Atk53]. The study demonstrates that highly motivated users have a tendency to recall interrupted tasks better as compared to recalling completed tasks. On the other hand, less motivated people show a tendency to recall completed tasks better than recalling the interrupted task. Overall, his findings suggest that the ability to recall the primary task after handling interruptions varies across people. A few years later, similar findings were reported by Bernard Weiner [Wei65].

Joslyn and Hunt proposed "The Puzzle Game" – an empirically validated test that can quantify the performance of users for multitasking [JH98]. Through a series of experiments, the authors showed that there are differences in people's ability to make rapid decisions for task switching. They suggested that people who are good in making quick decisions can be identified through testing their psychological characteristics, which can be captured by their puzzle game.

Brause and Wickens in [BW86] investigated the individual differences in sharing time between tasks in a multitasking environment. Their analysis show that there are differences in time-sharing ability of individuals; these are linked to their potential to process information. Moreover, their findings suggest that people have different strategies for time-sharing, which introduces differences in individuals' multitasking ability.

Moreover, some studies have demonstrated that people show significant differences in their cognitive style with respect to multitasking [Hus87, CCM90]. Studies have demonstrated that the performance in carrying out an interrupted task is affected by users'

anxiety [Hus87] and arousal [CCM90] levels. However, the level of anxiety and arousal varies across people [Hus87, CCM90].

2.5 Interruptibility Management

2.5.1 Definition

As discussed in the previous sections, interruptions are the part of our everyday life and often cause disruption that adversely affects the ongoing task and our psychological state. *Interruption management* is a process that combines technology, practices and policies to build solutions for controlling interruptions from seeking users' attention at inopportune moments. The key goal of an interruptibility management system is to help people to effectively perform their primary task and make computing devices calm by unobtrusively mediating interruptions.

2.5.2 Attentiveness and Receptivity to Interruption

In [PdOKO14] *attentiveness* is defined as the amount of attention paid by users towards their computing device for a newly available interruption task. However, attentiveness does not consider the response of the user to the interruption, which can be either positive (i.e., accepting the interruption) or negative (i.e., rejecting the interruption). On attending an interruption users get subtle clues about different features of interruptions such as a brief description of the content, sender and urgency of the interrupting task.

On the other hand, *receptivity* is defined as the process of making a decision about the way in which the user is willing to respond to an interruption by analysing its clues. In [FYB+10] Fischer at el. argue that users' receptivity to an interruption not only encompasses their reaction to that specific interruption but also their subjective experience of it. However, users' receptivity varies with the context as it accounts for their negotiation to handle interruptions in different contexts.

In this thesis we will consider both attentiveness and receptivity as key dimensions for the design of intelligent notification mechanisms.

2.6 Interruptibility Management in Desktop Environments

Interruptibility management has attracted the interest of HCI researchers well before the advent of mobile devices. However, interruptions received on the desktop have very specific characteristics. In fact, because of their very nature desktops are situated in a constant environment and the user's physical context (such as location, activity and surrounding people) does not always change, whereas mobile devices are carried by users almost everywhere, which makes the mobile user's physical context very dynamic. Therefore, interruptibility management for desktop environments is in a sense less complex.

2.6.1 Interruptibility Management by Using Wizard of Oz Approach

In 1993, Horvitz et al. started the Lumiere project [HJH99] with the goal of building a probabilistic model that can make predictions about the user's attention under uncertainty. They employed the Wizard of Oz technique to conduct a series of experiments.[1] The data obtained from these experiments was later fed to the manually constructed Bayesian networks to predict the users' need by using different variables such as their background, actions and queries. The constructed Bayesian models were used to obtain a probabilistic distribution for the degree of users' attention [HKPH03].

Opportune moments to interrupt people can be better predicted by considering a wide range of context information. For the first time, in [HFA+03] Hudson et al. explored the

[1] The Wizard of Oz technique is an experimental mechanism for the observation of a user operating an apparently fully functioning system whose missing services are supplemented by a wizard (i.e., a hidden observer) [DJA93]. Here, the user is led to believe that the computer system is fully operational and is not aware of the presence of the wizard. The wizard observes the user through a dedicated computer system connected to the observed system over a network. When the user invokes a function that is not available in the observed system, the wizard simulates the effect of the function.

possibility of detecting estimators of human interruptibility by using sensors in order to enhance computer mediated communications in work settings. They chose a Wizard of Oz approach which enables them to simulate a wide range of sensors. Using the data collected from these sensors they constructed a series of predictive models that achieved a level of the accuracy in the 75-80% range. Their study provided evidence that sensor information can be effectively used to estimate human interruptibility. Later, Fogarty et al. improved these models by exploiting additional sensor readings [FHA+05]. In [FHL04, FKA+05], the authors further examined the robustness of the proposed sensor-based approach by conducting experiments in a real world scenario.

2.6.2 Interruptibility Management by Exploiting Task Phases

In 1986, Miyata and Norman argued that people are less prone to perceive interruptions as disruptive in some phases of a task compared to other phases [MN86]. This suggestion was later investigated by Czerwinski et al. [CCH00b] in 2000 and by Cutrell et al. [CCH01] in 2001. Both studies confirm the insights of Miyata and Norman that perceived disruption varies when an interruption arrives at different phases of the task. Moreover, Czerwinski et al. [CCH00b] found that interruptions are less disruptive when they arrive at the completion of a primary task.

On the other hand, in [CCH01], Cutrell et al. found that interruptions arriving at the beginning of a primary task are perceived as less disruptive than interruptions occurring at other phases of a task execution. They suggested that interruptions should be queued and delivered when a user is switching tasks, rather than delivering them immediately. They designed an interface for instant messaging that constantly monitors user actions and infers the different phases of the task completion level as well as the moment when the users switch from one task to another. They also suggested that this information can be used to deliver instant messages at opportune moments.

Horvitz et al. devised the term "bounded deferral" [HAS05], which utilises the concept of deferring the interruption if the user is busy and determines the time until which the

interruption should be held from being delivered in order to minimise the disruption cost without losing the value of information due to the delay. The interruption is deferred until a maximum amount of time that is pre-specified by users (known as maximal deferral time) and after this time has elapsed, if they still remain busy, the alert is delivered immediately. They investigated the *busy* versus *free* situations for 113 users over a period of two consecutive days (i.e., a day for each situation). Participants were provided with a "Busy Context" tool that allows them to define when they were busy or free. The analysis of the data showed that users switch from busy to free situations in approximately 2 minutes. Moreover, they demonstrated that medium and low urgency emails can be deferred for three and four minutes respectively. They suggested that bounded-deferral policies can reduce the level of interruption while allowing users to be aware of important information.

As discussed earlier in this chapter, Adamczyk et al. demonstrated through a controlled experiment that the breakpoints within a task are the opportune moments to deliver interruptions [AB04]. Later, in [AIB05] they proposed a system that can automatically infer the breakpoints in tasks and exploit this information to deliver interruptions. In order to build this system they leveraged the findings of [Bea82, HL93, NTS02], which suggested that users' mental workload has a statistical correlation with the size of the pupil. They validated this in a human-computer interaction environment by showing pupillary response aligns with the changes in mental workload [IZB04, IAZB05]. They built a system that uses a head-mounted eye-tracker for measuring users' pupil size. Finally, they showed that their system was able to infer mental workload for two tasks (route planning and document editing) with an average error of 2.81% and 2.3% respectively.

In 2007, Iqbal and Bailey investigated the feasibility of inferring three types of breakpoints (coarse, medium and fine) during the execution of a task without using any supplementary hardware resources [IB07]. Data was collected from several participants in the form of event logs and screen interaction videos while they were performing the task. They recruited observers to watch the participants performing the task in order to identify

and label breakpoints, their type and explanations for their choice. Using the suggestions reported by Fogarty et al. in [FHL04, FKA+05], they identified a series of features (such as "switched to another document", "closed an application", "completed scroll", and so on) by analysing observers' explanations for breakpoints and event logs. Based on these features they constructed statistical models, which were able to predict each type of breakpoints with an average accuracy of 69% to 87% (i.e., the percentage of correctly identified breakpoints over the total predicted breakpoints). For each type of breakpoint different features were computed and also different models were constructed.

In [IB06] Iqbal and Bailey investigated whether the characteristics of task structure can be used to predict the cost of interruption. They posited that the cost of interruption is measured only by the resumption lag (i.e., the time taken by users to resume back to their primary tasks after handling an interruption). More specifically, they tried to test whether interruptions arriving at boundaries of subtasks have low cost of disruption, by using the task structure rather than using head mounted cameras to measure pupil size as they did in [IB07]. Here, *task structure* indicates the subtasks and boundaries within a task decomposition and the characteristics of the task structure refers to the depth of task decomposition, types of subtasks and mental carryover [CNM83]. They evaluated their approach by conducting experiments on a set of primary tasks. In order to collect the data for estimating the resumption lag on receiving interruptions, participants were interrupted at various boundaries of the task execution. Finally, they constructed a statistical model by using characteristics of task structure for predicting the cost of interruption. Their results suggest that their approach could predict the cost of interruption with a 56–77% accuracy for all tasks.

2.6.3 On-the-fly Inference of Interruptibility

All of the studies discussed above were on an *offline* recognition and construction of models to identify opportune moments for triggering interruptions. Lilsys [BMT04] and BusyBody [HKA04] were the first attempts at designing solutions for on-the-fly inference

of interruptibility. Both systems were built with custom hardware merely for research projects and focused on interruptions in an office environment.

Lilsys [BMT04] uses ambient sensors to detect user's unavailability for telecommunication. The system consisted of sensors including sound and motion sensors, phone and door usage inference through the attached wires and keyboard/mouse activity inference. Moreover, it allowed users to manually register their unavailability (if they wanted to) and turn on/off the sensing whenever they wanted. Data is collected passively and an inference about the user's presence and availability is made on detecting a change in any sensor event. Lilsys uses the data from phone, keyboard, mouse, motion and sound detectors in order to predict the user's presence. On the other hand, the unavailability is predicted by exploiting the data from sound, phone, and door sensors. Both predictors are based on a simple Decision Tree model. The system was deployed in an office for around seven months. Participants reported the qualitative improvement in the interruptions but not much reduction in the quantity of interruptions.

BusyBody [HKA04] is based on an initial training phase during which the system asks users about their interruptibility at random times and it also continuously logs the stream of desktop events, meeting status and conversations. Then, on completion of the training phase it uses the collected information to build predictive models based on Bayesian networks for inferring the cost of interrupting users (i.e., the level of perceived disruption) in real-time. Through a small-scale study, the authors demonstrated that BusyBody is able to make predictions about the cost of interrupting users with an accuracy of 70%-87%.

Both Lilsys and BusyBody represent valuable applications of machine learning algorithms for exploiting the contextual information to predict interruptibility. Similarly, Iqbal and Bailey proposed OASIS [IB10] – a system that detects the breakpoints in users' activity independent of any task by exploiting the streams of application events and user interaction with a computer. This information is used to determine the notification scheduling policies on-the-fly and to deliver notifications accordingly.

2.7 Interruptibility Management in Mobile Environments

When mobile devices first appeared, they were used merely for calling and messaging purposes. Later, with the advent of sensing capabilities, these devices have graduated from calling instruments to intelligent and highly personal devices performing numerous functions salient to users' daily requirements. This has provided opportunities to mobile applications to connect users to different information channels and deliver them updates in real-time about a much larger variety of events, ranging from messages to traffic alerts and advertisements.

2.7.1 Understanding Users' Perception Towards Mobile Notifications

Studies have shown that people receive a much higher number of notifications everyday on their mobile phone compared to desktop notifications [PCdO14]. Due to the pervasive nature of mobile phones notifications arrive at anytime and anywhere, which makes mobile interruptions more obnoxious compared to desktop interruptions. Thus, managing interruptions in mobile settings has also become a more complex and important task.

Even though people report that notifications are disruptive, they still like to continue receiving them in order to keep themselves aware of newly available information automatically instead of manually checking [IH10]. People tend to use some simple strategies of their own in order to manage interruptions. In a study with several participants [CT15], Chang and Tang found that people mostly manage interruptibility through the ringer mode of mobile phones. Findings of another study [WMW15] suggest that only a few people tend to change notification settings for individual applications, such as stopping notifications from a specific application. Similar findings were reported by Lopez et al. in [LTCD15]. The authors of the study also suggested that people want a fine-grained

control for how, when and which notifications are delivered to them, which is not present in any mobile platform.

Sahami et al. ran a large-scale study [SSHD+14] to understand how users perceive mobile notifications. They collected around 200 million notifications from more than 40,000 users over a period of 8 months. Their results show that even though people deal with many notifications each day, most notifications are still viewed within a few minutes of their arrival. Additionally, by collecting the subjective feedback from mobile users, the authors show that users assign different importance to notifications triggered by applications of different types (i.e., categories). Their findings suggest that people value notifications triggered by messaging applications as well as notifications that include information about people and events.

In another recent study [PCdO14], Pielot et al. found that most mobile notifications, received by people, are about personal communication, which are triggered by messenger and email applications. Also, the authors found that users are not always interested in the information pushed by proactive services. Regardless of the ringer mode of phone, notifications are viewed within a few minutes of arrival. By collecting the subjective feedback from mobile users, they found that personal communication notifications are responded to quickly because of social pressure and shared indicators of availability (such as "the last time the user was online") provided by communication applications. Moreover, their findings suggest that, although, personal communication notifications make the users feel connected with their social links, the increasing number of such notifications becomes a source of negative emotions and stress.

In [FGB11] Fischer et al. investigated the behaviour of users on receiving notifications from specific categories of application. They found that the reaction to notifications is a function of their importance. According to a field study with 11 co-workers by Fischer et al. [FYB+10], the subjective importance of SMS-related notifications depends on how interesting, entertaining, relevant, and actionable a message is. If apps that are not perceived as useful keep sending notifications, users become annoyed and consider

deleting those applications [FEW12]. This suggests that there is a clear need for reducing interruptions caused by wrongly timed and irrelevant notifications.

2.7.2 Predicting the Cost of Interruption

In [HKS+05] Horvitz et al. proposed the first interruptibility management tool for predicting the cost of being interrupted by incoming calls. The system consisted of two Bayesian network models that were trained for predicting users' interruptibility and attendance to meetings that appear on their electronic calendar. The prediction results of these models were used to compute the cost of interrupting a user through phone calls. The models were trained by using the sensed activity and calendar properties. It is worth remarking that this system was designed solely for managing phone calls, which were the only interruptions generated by mobile devices at that time.

In [RDV11] Rosenthal et al. discussed a personalised approach for predicting the cost of a mobile interruption. They conducted a survey to understand mobile phone users' preferences and interruption cost in different situations. Their results suggest that the cost of interruptions vary across users. For instance, a user might not have any problem in receiving interruptions at work, while another user might consider these as a significant disruption. Therefore, according to their findings, interruptibility management models should be personalised. Moreover, they conducted an experiment to compare the usability and accuracy of different sampling techniques that are used in the training phase of a prediction model for automatic setting of the phone ringer. More specifically, they recruited participants to train a preference classifier for two weeks. Participants were asked to express their preferences for a mobile phone's volume in different situations. Using this training set, the authors constructed personalised predictive models that are used to automatically turn on/off the volume of the ringer. The resulting system was tested for two weeks. Participants were asked to report the correctness of the model by filling in a daily questionnaire.

Participants were randomly assigned to one of the sampling techniques: random sampling, uncertainty-based sampling [LC94], decision-theoretic sampling [KH08] and cost estimation sampling (i.e., the technique proposed by the authors). Their results show that the proposed technique had a significant effect on the number of surveys presented to the users. They found that, by using their model, 7 out of 10 participants reported very high accuracy with few or no errors made by the system for automatically turning the phone's volume on or off in order to avoid unwanted interruptions.

2.7.3 Interruptibility Management by Using the Transition between Activities

In [HI05] Ho and Intille explored the use of transitions between physical activities for delivering mobile notifications. More specifically, they conducted an experiment to compare users' receptivity to mobile notifications triggered at times corresponding to activity transition and at other times. Their study was based on the hypothesis that physical activity transition indicates "self interruption" as users switch to another one after its completion. Such a moment might lower the user's resistance to an interruption. The authors customised a few PDAs by adding two wireless accelerometers in each in order to capture users' physical movement. After using temporal smoothing on activity data, they captured four types of transitions: sitting to walking, walking to sitting, sitting to standing and standing to sitting. Their results show that interruptions delivered at a time corresponding to an activity switch are judged more positively compared to interruptions delivered at random times.

In another study [FGB11] Fischer et al. investigated the use of naturally occurring breakpoints during users' interaction with mobile phones as opportune moments to deliver mobile notifications. Through an experience-sampling study the authors asked users to provide feedback about their context in terms of their interactions with the phone. These notifications were delivered at random times or after the user has finished a call or finished sending/reading an SMS. Their results suggest that the participants reacted faster to

notifications when these are delivered right after finishing a phone call or sending/reading a text message. However, the authors could not show whether the subjective experience improves, perhaps because the artificial notifications created extra work.

In [ONN+16] Okoshi et al. studied the use of breakpoints within users' interaction with a mobile phone for delivering notifications in order to reduce interruptions and improve users' experience. The authors developed Attelia – a system for detecting breakpoints in users' interaction with mobile phones and to defer notifications until such a breakpoint occurs. Attelia detects breakpoints during the user's interaction with a mobile phone in real-time, by using the sensors embedded in the phone. Attelia monitors users' interaction with applications and exploit this information in order to detect breakpoints. They used the NASA-TLX questionnaire [HS88] to quantify participants' subjective cognitive load. The authors first conducted a controlled study with 37 participants to evaluate the system. Their results demonstrated that, by delivering notifications at breakpoints, the cognitive load of users who showed greater sensitivity for interruptive notification timings is reduced by 46% compared to delivering notifications at random times. Later, they conducted an "in-the-wild" study with 30 participants for 16 days in order to validate the system in real-world settings. The results of this "in-the-wild" study suggest that Attelia could reduce 33% of the cognitive load by delivering notifications at detected breakpoints. Moreover, the notifications delivered at breakpoints received a quicker response from users.

2.7.4 Interruptibility Management by Using Contextual Data

Today's mobile phones are laden with sensors that are able to monitor various context modalities such as physical movement, sound intensity, location and colocation with other Bluetooth devices (i.e., colocation with users). Some previous studies have showed the potential of mobile phones in providing data that can be used to infer not only numerous aspects of users' physical behavioural patterns [LML+10, LXL+11, MPM14] but also their health and emotional states [RMM+10, LMRF+12, LCC+11, LCLP12, LLLZ13, CM15].

Scientists have used various aspects of the physical context of users, captured via mobile sensors, in order to construct machine learning-based models for predicting interruptibility of users. For example, Pejovic et al. developed InterruptMe [PM14b] – an interruption management library for Android-based mobile devices. InterruptMe uses a mixed method of automated smartphone sensing to collect contextual information and experience sampling to ask users about their interruptibility at different moments. This information is exploited by InterruptMe to construct intelligent interruption models based on a series of machine learning algorithms for predicting the user's interruptibility at the current moment. In order to evaluate InterruptMe, the authors conducted a two-week study with 20 participants and gathered users' *in-the-wild* context, including their activity, location, the time of day, emotions and engagement with an ongoing task. Their results show that opportune moments for interruptions can be predicted with an average accuracy of 60% and the reported sentiments towards notification can also be predicted with a precision of 64% and a recall of 41%. Moreover, they found that the time taken by users to respond to notifications can be predicted accurately. Finally, they demonstrated that the online learning approach can be used to train models well enough to start making stable predictions within a week. The experiments also revealed that such moments cannot be considered in isolation and that users' sentiment towards an interruption depends on the recently experienced interruption load.

In [PdOKO14] the authors conducted a survey with 84 users to understand how people perceive shared indicators of availability (such as "the last time the user was online") provided by messaging applications. Their results show that these indicators create a social pressure on users to respond to the messages but people still see a great value in sharing their attentiveness. However, the authors argued that the shared indicators of availability by messaging applications are weak predictors of recipients' attentiveness. Instead, machine-computed prediction of recipients' attentiveness should be used as a more reliable source. In order to validate their proposed approach, the authors conducted an experiment with 24 participants for a period of two weeks. They developed a mobile

application that logs information about users' context and their actual attentiveness, including application name and the arrival time of messages, elapsed time between the arrival time and the time when the message was actually read, launching and closing times of messaging applications, phone lock/unlock times and the phone's ringer mode (silent, vibration and sound). Using this data they computed 17 features and ranked them based on their entropy. They demonstrated that by using only the top seven features, a machine learning algorithm can construct a model that can predict users' attentiveness with an accuracy of 70.6%

In [DP15] Dingler and Pielot argued that bounded-deferral strategies (i.e., strategies for deferring notifications up to a certain time period in order to reduce disruption caused by it) do not work if users are busy for long time periods. They suggested that a notification might lose its value if it is delayed for too long. Therefore, notifications must only be deferred if the phases of inattentiveness are brief. Based on their hypothesis they conducted a study to investigate whether users' attentiveness to mobile phone notifications can be predicted by using contextual information. Through a passive and continuous sensing approach, they collected phone-usage data from 42 participants for a period of two weeks. They demonstrated that users' attentiveness can be predicted using mobile phone usage with an accuracy of 80%. Their findings show that users are attentive to mobile notifications for around 12 hours in a day. Also, the periods of users' inattentiveness to mobile notifications are often very short (i.e., 2-5 minutes).

Another study [Pie14] explored the use of phone usage activity and contextual information for predicting users' attentiveness to calls. In order to collect data, the authors developed an application that temporarily mutes the ringer by simply shaking the phone. They logged anonymous data from 418 users corresponding to more than 31000 calls mapped with recipients' context at the time of call arrival. They collected data which is available through the open API calls of the Android platform, such as physical activity, ringer mode, device posture and time since last call. They demonstrated that these features can be used to predict call availability with an accuracy of 83.2%. Moreover, they

showed that a personalised model training approach can increase the average accuracy by 87%. By using only the top five features (including last time the ringer mode was changed, last time the screen was locked/unlocked, current status of screen lock, last time the phone was plugged/unplugged from charging, time since last call) call availability can be predicted with an accuracy of 79.62%.

On the other hand in [GJ09], Grandhi et al. suggested that by just relying on sensor based knowledge, interruption management systems often fail to infer whether the interruptions are disruptive or not because they do not take into account the sender of an interruption and the information that is being sent. The authors propose a theoretical framework for interruptibility management that takes into account information about notification sender and content. However, the authors do not support their claim through a deployment in a mobile setting.

2.7.5 Interruptibility Management by Filtering Irrelevant Information

Previous studies have shown that users are willing to tolerate some interruptions from notifications so that they do not miss any important information [IH10]. However, their willingness is, in a sense, exploited by mobile applications as these trigger a plethora of notifications continuously [PCdO14]. Given the potentially large number of notifications, users mostly dismiss (i.e., swipe away without clicking) those that are not useful or relevant to their interests [SSHD$^+$14]. Some examples of such notifications are promotional emails, game invites on social networks and proactive suggestions by applications (such as traffic updates). At the same time, past studies have shown that users get annoyed by receiving irrelevant or unwanted notifications that could result in the uninstallation of the corresponding application [FEW12]. This suggests that learning about the importance of notifications for the user may be a feasible approach to delivering only relevant notifications to them.

In [FYB+10] Fisher et al. investigated the effects of notification content and the time of its delivery on the user's receptivity to that notification. In order to understand the role of notification content, they recruited 11 participants who were asked to rate their interest in the given 28 categories of content on a 7-point Likert scale. For each participant, the content types rated (by the same participant) as top 3 were considered as "good content" and the lowest three were considered as "bad content". Moreover, participants were asked to specify the time window during which they were interested in receiving notifications of these types. Participants received six notifications (three each of good and bad content) every day. These notifications were delivered at three opportune times and three inopportune times. Their results show that users' receptivity for good content was significantly higher than their receptivity for bad content. However, there was no significant differences in users' receptivity at opportune and inopportune times. This might suggest that users' receptivity is associated with notification content rather than the time of delivery. Furthermore, the participants were asked to fill a daily survey. The survey asked them to provide a rating for the perceived interest, entertainment value and actionability required to handle the notification. This data was used to understand the properties of notification content that make people receptive to it. Their findings suggest that users' interest, entertainment, relevance and actionability significantly influence their receptivity.

2.8 Limitations of the Existing Literature and Scope of this Thesis

Interruptions are an inevitable part of our daily life. As discussed in this chapter, the effects of disruption caused by interruptions occurring at inopportune moments have been studied thoroughly in the past. Studies have clearly demonstrated that interrupting users engaged in tasks has a considerable negative impact on task completion time [CCH01, CCH00a, MBDT02], error rate [Lat98], and even emotional state [AB04, BK06]. However,

studies have also provided evidence that users are willing to tolerate some disruption in return for receiving notifications that contain valuable information [IH10].

Since the era of desktops, managing interruptions has been a key theme in Human-Computer Interaction research [HJH99, HKA04, AIB05, IB06]. With the advent of mobile technologies, the problem of managing interruptibility has become even more pressing as users can now receive notifications anywhere and at anytime. For this reason in the past years we have witnessed numerous research efforts directed at managing interruptions in mobile environments [HI05, IH10, PM14b, ONN+16]. Most of the work on mobile interruptibility emphasises the exploitation of task phases [HI05, FGB11, IH10] and users' context [PM14b, PdOKO14, DP15] to infer the right time to interrupt.

Studies have shown that notifications are considered more positively and received a faster response when delivered while a user switches between two activities, such as sitting and walking [HI05, FGB11]. On the other hand, studies have also demonstrated that machine learning algorithms can learn about users' interruptibility by exploiting passively sensed contextual information and interruptible moments reported by users [PM14b]. Furthermore, sensed data can be used to compute features, such as last notification response, phone's ringer mode and proximity sensor, which are effective predictors of the notification response time.

However, existing studies do not completely address the problem of characterising users' attentiveness and receptivity to notifications. The key reason behind this is that there is still a lack of understanding concerning the factors influencing users' attentiveness and receptivity to mobile notifications in different cognitive situations. Moreover, until now, studies ignore the role of content (i.e., who is the sender and what information is being sent) for modelling interruptibility [GJ09]. Therefore, existing interruption management systems have not shown remarkable performance in terms of the prediction of opportune moments for delivering information [PM14b, ONN+16]. A few past studies have also incorporated aspects of content for studying interruptibility [SVV03, FYB+10]. In [SVV03] Speier et al. have shown the relevance of content with the current task

when predicting the interruption caused by the pushed information. On the other hand in [FYB+10], Fischer et al. claims that interruptibility can be determined by the user's interest, entertainment and actionability in the notification content. In general, there is very little work about the design of an intelligent notification mechanism based on a user's preferences about "from whom (sender)", "what (type of information)", and "where (user's context)" to receive mobile notifications. This thesis represents an initial step towards realising a complete intelligent notification mechanism by considering all the three factors together.

This chapter has discussed the state-of the-art of the area of human interruptibility. We identified four key limitations of the existing literature that are addressed in this thesis:

(i) There is still a lack of understanding about the role of users' emotional states and their engagement with a primary task in influencing a user's interaction with notifications. In Chapter 3 we present two studies for bridging this gap. It is worth noting that the use of these cognitive features for modelling interruptibility is out of the scope of this thesis. Indeed, it is not possible to get a continuous stream of cognitive contextual information of users as this is queried via ESM and it would not be easy for users to respond to questionnaires very frequently. For this reason, we have collected such data only for investigating the association of cognitive context with interruptibility. The use of cognitive context for the design of predictive interruptibility models is not in the scope of this thesis.

(ii) Almost all mobile interruptibility management studies have focused on exploiting physical contextual information for building interruptibility models [HI05, PM14b, ONN+16]. On the other hand, the literature suggests that users' interruptibility also depends on the information about the sender and the content of notifications [SVV03, FYB+10]. In Chapter 4 we present the design and implementation of an interruptibility model that exploits information about users' context, sender and content of notifications to learn about their behaviour for interacting with notifications.

(iii) Some studies have shown that not all notifications that users receive are useful or relevant for them [FYB+10, SSHD+14]. However, the existing literature remains limited in showing how an interruptibility management system can learn which notifications are not at all useful/relevant for users in specific situations. In Chapter 5 we address this challenge for automatic learning of users' preferences through text mining and machine learning algorithms.

(iv) Another key limitation of the existing literature is that none of the previous studies have focused on understanding users' behaviour on receiving cross-platform notifications, which are delivered on multiple devices at the same time. In other words, there is a lack of understanding about features that determine users' receptivity to such notifications on a specific device in a given context. In Chapter 6 we address this challenge by designing, implementing and evaluating a solution for intelligent notification in multi-device environments.

CHAPTER 3

UNDERSTANDING PEOPLE'S ATTENTIVENESS AND RECEPTIVITY TO MOBILE NOTIFICATIONS

3.1 Overview

Mobile notifications are extremely beneficial to users. However, at the same time, they are a cause of potential disruption, since they often require users' attention at inopportune moments for them. Indeed, previous studies have found that interruptions at inopportune moments can adversely affect task completion time [CCH01, CCH00a, MBDT02], lead to high task error rate [BKC00] and impact the emotional and affective state of the user [AB04, BK06]. Also, users get annoyed when they receive notifications presenting information that is not useful or relevant to them in the current context [CCH00a]. This tension is exacerbated by the fact that individuals have to deal with a plethora of notifications in a day, some of which are disruptive [PCdO14]. At the same time, studies have shown that even though people perceive disruption through notification alerts, they are still receptive to these notifications in order to avoid missing any important information [ORMR12].

Previous studies have shown that the user's receptivity to a notification is determined by: (i) physical context [PM14b]; (ii) how interesting, entertaining, relevant and action-

able its content is for the user [FYB+10]; (iii) the type of application that triggers it – communication applications are considered as the most important [SSHD+14]; (iv) time criticality and social pressure [PCdO14]. All of these studies have mainly focused on exploring the role of physical context and notification content in influencing the user's receptivity to mobile notifications. However, there is still a lack of understanding concerning the impact of cognitive and notification design factors on users' receptivity. This might also be the key reason for the failure of most interruptibility management systems to achieve a very high accuracy in predicting the opportune moments for interrupting users.

In order to bridge this gap, we conduct two in-situ studies to investigate the factors impacting users' perception towards, attentiveness and receptivity to mobile notifications. The first study [Section 3.2] examines the role of the ongoing task engagement and notification design. In the second study [Section 3.3], we examine the impact of emotional states on users' attentiveness and receptivity to mobile notifications[1].

The key contributions of this chapter are investigation and characterisation of:

- the impact of a notification's alert modality on the user's ability to perceive a notification alert;
- the impact of the alert modality, sender-recipient relationship, presentation of a notification, the ongoing task type, completion level and task complexity on the response time;
- the impact of the sender-recipient relationship, and the ongoing task's type, completion level and complexity on the perceived disruption;
- the role of the sender-recipient relationship, notification content and the perceived disruption on the user's decision to accept or dismiss a notification;

[1]The second study presented in this chapter was performed in a collaboration with another PhD student (Fani Tsapeli). Most of the work (including experimental design, application development, data collection, data preprocessing and correlation analysis) was done by me. However, Fani Tsapeli provided support to perform the causality analysis by using her framework presented in [TM15].

- the association between users' interaction with notifications and changes in emotional states;
- the causal relationships between users' emotional states and notifications interaction.

3.2 Understanding the Role of Ongoing Task Engagement and Notification Design

In this section we will discuss the first study which is aimed at understanding the impact of ongoing task engagement and notification design on users' reactions to notifications. More specifically, we investigate the impact of the alert modality, sender-recipient relationship, presentation of a notification, the ongoing task type, completion level and task complexity on the seen, decision and response times.

3.2.1 Data Collection

In order to investigate the impact of ongoing task engagement and notification design on the user's receptivity to mobile notifications, we conducted an in-situ field study. More specifically, we developed *My Phone and Me* (see Figure 3.1) – an Android experience sampling method (ESM) application that collects information about *in-the-wild* notifications, users' interaction with them in natural situations (while they are performing their day-to-day activities), and the physical and cognitive context information.

The My Phone and Me application uses Android's Notification Listener Service [And16a] to access notifications, and Google's Activity Recognition API [GAR16] and ESSensorManager library [LRMR13] to obtain the context information. Table 3.1 lists the groups of features captured by the application. It is worth noting that the collected context data has not been used for the analysis presented in this work. Instead, this data is explored in the study presented in Chapter 5.

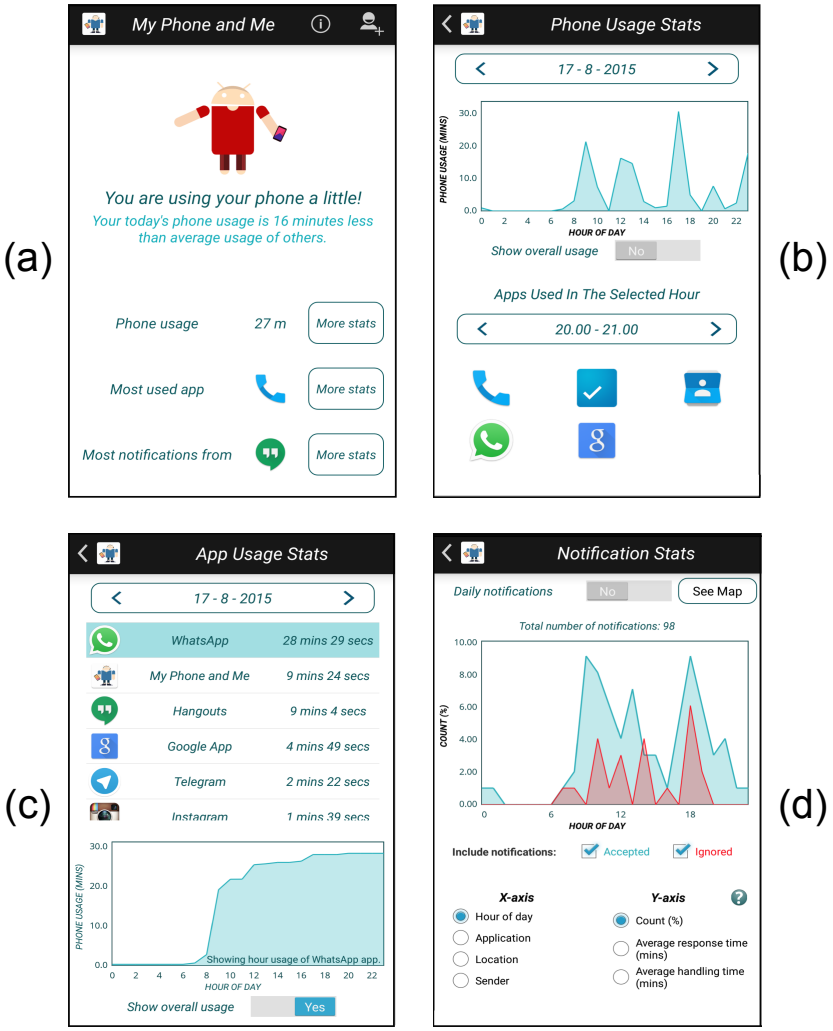


Figure 3.1: My Phone and Me application: (a) main screen, (b) phone usage statistics, (c) application usage statistics, (d) daily notifications.

In order to capture a better picture of people's interaction with mobile notifications, the My Phone and Me application logs three time measurements (as shown in Figure 3.2) for each notification: the time of notification arrival (a), the time when the notification is seen (b), and the time when the user accepted (c1) or dismissed (c2) the notification. Note that in order to detect the moment when a notification is seen, we use the unlock event of the phone and assume that all newly available notifications in the notification bar are seen when the user unlocks the phone. In case a notification arrives when the user is already using the phone (i.e., the phone is unlocked), the seen time of this notification would be computed as zero. More specifically, for our analysis we compute the following three time measurements for users' interaction with notifications:

Group	Features
Time	Arrival, seen and the removal time of a notification.
Notification response	Whether the notification was accepted (i.e., clicked) or dismissed.
Notification details	Sender application and the title of a notification.
Alert type	Signals used by a notification to alert the user: sound, vibrate, and flashing LED.
Context data	Physical activity, location, presence of surrounding sound, WiFi connectivity, proximity to the phone, surrounding light intensity. This data is collected on arrival and removal of a notification from the notification bar.

Table 3.1: Description of the features of the My Phone and Me dataset.

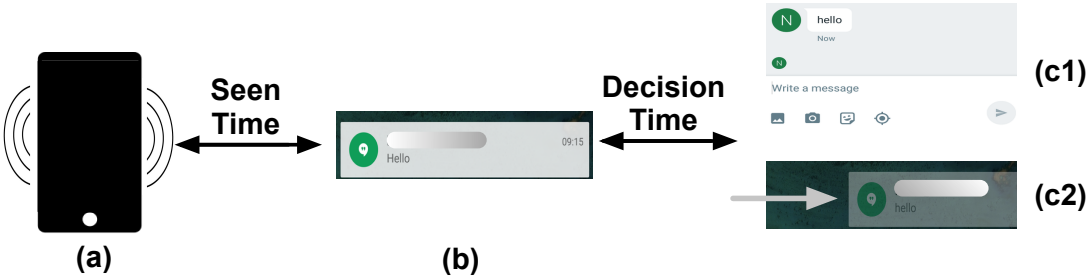


Figure 3.2: The three time measurements of a notification captured by the My Phone and Me application. The time of notification arrival (a), the time when a notification is seen (b), and the time when the user accepted ($c1$) or dismissed ($c2$) a notification. The time difference between (a) and (b) is *seen time* and the time difference between (b) and ($c1$ or $c2$) is the *decision time*.

- Seen time (ST) – time from the notification arrival until the time the notification was seen by the user.
- Decision time (DT) – time from the moment a user saw a notification until the time he/she acted upon it (by clicking, launching its corresponding app or swiping to dismiss).
- Response time (RT) – time from the notification arrival until the time the notification is reacted upon by the user. It is the sum of seen time and decision time.

Moreover, to infer the user's response to a notification, the My Phone and Me application checks whether the application that triggered the notification was launched after the removal time of that notification. It is possible that some notifications are dismissed because they do not require any further action. For this reason, the My Phone and Me

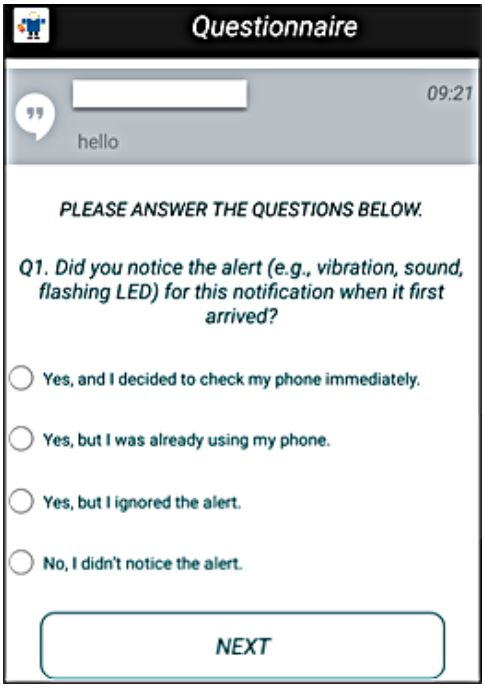


Figure 3.3: The screenshot of a questionnaire triggered by the My Phone and Me application.

application captures seen time and uses the difference between seen time and removal time to understand how long it takes for the user to read and react to a notification.

To collect subjective data from users, the My Phone and Me application triggers four questionnaires in a day. A questionnaire is triggered only when a notification is handled; it contains questions about why the notification was clicked or dismissed by presenting a screenshot of that notification (see Figure 3.3). The application triggers a questionnaire for a randomly selected notification in every 4 hour time window between 8.00 am and 8.00 pm and the last questionnaire at a random time between 8.00 pm and 10.00 pm. The application did not trigger any questionnaire after 10pm so that the participants do not feel annoyed at responding to the surveys late at night. The application automatically used the local time zones because it relies on the phone's time. Moreover, if the user is busy, the questionnaire can be dismissed by simply swiping it from the notification bar and no questionnaire is shown to the user for the next 30 minutes.

A questionnaire comprises seven multiple-choice and two free-response questions. The list of questions and their options are shown in Table 4.2. Since the application asks

Question	Options
Did you notice the alert (e.g., vibration, sound, flashing LED) for this notification when it first arrived?	(i) Yes, and I decided to check my phone immediately. (ii) Yes, but I was already using my phone. (iii) Yes, but I ignored the alert. (iv) No, I didn't notice the alert.
How did you handle the notification when you first saw it?	(i) I decided to immediately click it. (ii) I decided to dismiss it because it didn't require any further action. (iii) I decided to dismiss it because it was not relevant or useful. (iv) I decided to return to it later. (v) Other (descriptive).
Select all factors that made you decide to click/dismiss the notification.	(i) The sender is important. (ii) The content is important. (iii) The content is urgent. (iv) The content is useful. (v) I was waiting for this notification. (vi) The action demanded by the sender does not require a lot of effort. (vii) At this moment, I was free. (viii) Other (descriptive).
What best describes your relationship to the sender?	(i) Partner (ii) Immediate family (children, parents) (iii) Extended family (nieces/nephews, cousins, aunts/uncles) (iv) Friend (v) Acquaintance (vi) Superior at work (vii) Colleague (viii) Subordinate at work (ix) Client (x) Service provider (xi) Sender is not a person (xii) Other relationship (descriptive).
Please describe what the notification was about.	Descriptive response.
Please describe what activity you were involved with when you received the notification.	Descriptive response.
When the notification arrived, I was:	(i) Starting a new task/activity. (ii) In the middle of a task/activity. (iii) Finishing a task/activity. (iv) Not doing anything.
The task/activity I was doing when the notification arrived was complex.	Five-level Likert scale rating between "strongly disagree" and "strongly agree".
I found the notification disruptive.	Five-level Likert scale rating between "strongly disagree" and "strongly agree".

Table 3.2: Questions and their options from the questionnaire triggered by the My Phone and Me app.

users to enter free form text for two questions, it could increase the time to respond to a questionnaire and may become a source of annoyance. Therefore, the application allows users to dictate the responses to these questions. These answers are then converted to text using Android's Speech Recognizer API [Spe16].

Recruitment of the Participants

The My Phone and Me application was published on the Google Play Store from 12th August 2015. It was installed by 74 participants without any monetary incentive. As shown in Figure 3.1, My Phone and Me tells the users about their addiction to the phone. It allows users to check statistics on their phone usage and interruptions. The application presents a visual representation of a user's phone activities based on different criteria, such as their hourly phone usage (Figure 3.1 C), hourly usage of individual applications (Figure 3.1 D) and how much they interact with notifications (Figure 3.1 B). We believe that displaying this information has a minimal interference with users' actual behaviour in interacting with notifications, but it provides a valuable functionality in order to make the users keep the application installed on their phones.

Ensuring Privacy Compliance

In order to ensure privacy compliance, the My Phone and Me application goes through a two-level user agreement to access users' notifications. Firstly, users have to give explicit permission as required by the Android operating system. Secondly, the application shows a list of information that is collected and asks for the user's consent. Moreover, the original content of a notification is shown to the user along with the questionnaire in order to avoid any recall bias in the data but we do not collect the notification content for privacy reasons.

3.2.2 Dataset

The data collection was carried out for around two months, during which we collected 19494 notifications and 611 responses to the questionnaire (comprising a set of nine questions listed in Table 4.2) from 74 users who installed the My Phone and Me application. Many users stopped responding to the questionnaires after a few days and some did not respond at all. Therefore, a subset of data was used for the analysis by considering

only the data of users who responded to at least 14 questionnaires. There were 20 users who satisfied this constraint. So, the final dataset consists of 10372 notifications and 474 questionnaire responses. Additionally, during the setup phase the application asked participants to enter their age and gender: in the final dataset there are 11 males and 9 females aged between 19 and 50 years old. However, users were not asked to provide any other demographic information.

As this work primarily uses the questionnaire responses for the analysis, we have compared the click rate (i.e., the percentage of notifications that are clicked out of the total number of notifications) of the overall notifications with the notifications that were linked to questionnaires. The click rate for overall notifications is 62.52%, and for notifications linked with questionnaires is 70.04%. Note that a notification is considered to be clicked either when it is clicked on the notification bar or when its corresponding application is launched directly.

3.2.3 Understanding Response Time

In this section we investigate the effect of different factors on the seen and decision time of a notification. The contributions of this section are summarised in Box 3.1.

The Role of Alert Modality in Perceiving a Notification Alert

A notification can alert the user by means of vibration, sound and/or flashing LED. In order to investigate how users perceive alerts with different alert modalities, we used the responses provided by the users for Q1 (*Did you notice the alert (e.g., vibration, sound, flashing LED) for this notification when it first arrived?*). According to our dataset, when the notifications (with which the questionnaires were linked) were triggered the user's phone was for 25.54% of the times in silent mode, 21.50% vibrate mode, 41.94% sound mode and 11.03% sound with vibrate.

The key findings of this section are:

- Users are aware of the notification alerts even when the phone is in silent mode. However, seen time is fastest when the phone is in vibrate mode and slowest for silent node.
- Notifications are seen fastest when the user is commuting and slowest when idle.
- User's attentiveness increases (reducing the seen time) with the increase in the complexity of an ongoing task.
- The decision time is higher for the notifications from less frequently contacted senders.
- High-priority notifications get quicker response.

Box 3.1: Key findings of the analysis for understanding response time.

Users reported that they missed notification alerts for 14.63%, 15.38%, 23.75%, 21.05% of times their phone was in silent, vibrate, sound, and sound with vibrate mode respectively. This provides evidence that when the phone is in silent mode users are still aware of the notification alerts.

What Factors Influence the Seen Time?

We investigate the role of alert modality, sender and the ongoing task type, complexity and completion level, in influencing the seen time of a notification.

It is worth noting that one of the key limitations of the in-the-wild study is that it becomes difficult to obtain an equal number of questionnaire responses from all users for each test category. For instance, there is a very small chance that the application triggers a questionnaire for each type of sender from the recipient's social circle. Furthermore, in practice, a user might not receive notifications from each of the sender types during the period of the study. Due to this reason our dataset remains unbalanced for all levels. Therefore, in case we use the mean of all samples for each condition from each user and then compare the means, we would have very unbalanced data (i.e., some levels with a

low number of samples compared to other levels). Moreover, we would also lose some information in the aggregation to user level means.

At the same time, we believe that the data collected from a user at different times might also be affected by the change in the user's physical context. For instance, response time to notifications might have a variation even when they are handled by a user while performing a similar task, but in different situations. Therefore, we do not use the aggregate of all samples for each condition from each user.

On the other hand, the examined variables (such as the response time of notifications) are not usually normally distributed. Therefore, we need to perform a non-parametric method (such as Kruskal–Wallis test) for analysing the means of samples.

A. The Impact of Alert Modality on Seen Time. In order to perform this analysis, from our dataset of 10372 notifications we ignore the notifications that arrived when the user was already engaged with the phone because we could not calculate the seen time of these notifications. This leaves us with 4929 notifications. A Kruskal-Wallis analysis of the seen time was carried out for each alert modality. The results show that the alert modality has an impact on the seen time of notifications, with $\chi^2(3, 4925) = 23.11$, $p < 0.001$. A Tukey post-hoc test (by setting α equal to 0.05) revealed that the seen time is statistically significantly higher for silent notifications (average 7.332 minutes). The seen time for the notifications alerting with vibrate only mode is the lowest (average 3 minutes and 21 seconds). Sound only and sound with vibrate notifications are the second (average 5 minutes and 57 seconds) and third (average 4 minutes and 50 seconds) most quickly seen by users. Quite interestingly, a recent 15-user study by Pielot et al. [PCdO14] also found that notifications tend to be seen more quickly when the phone is in the vibrate mode. Here, we confirm this finding, but also point to the above missed notification percentage in the silent mode (14.63%) and show that setting the phone to silent does not help in escaping interruptions.

B. The Impact of Ongoing Task Type on Seen Time. To investigate the impact of the ongoing task on the notification's seen time, we need to analyse the type of task that

users were involved with when the notification arrived. We classified the information that users provided through the ESM questionnaires about the ongoing task into the following six categories: work, communication, traveling, maintenance/personal, leisure and idle. The classification was done manually by two coders. Note that our app allows users to skip the step of providing the information on the question about their current task by selecting the "Prefer not to say" option. In such cases, we discard the entry from our analysis of the effect of the ongoing task on interruptibility.

A Kruskal-Wallis analysis of the seen time is carried out for each task type. The results show that the ongoing task type has an impact on the seen time of notifications, with $\chi^2(5, 217) = 10.92$, $p = 0.041$. A Tukey post-hoc test (by setting α equal to 0.05) reveals that the seen time is the lowest when the notifications arrive while the user is communicating (average 47 seconds) and highest while the user is idle (average 9 minutes and 30 seconds). Other task types do not have a statistically significant effect on the seen time of notifications and have an average seen time of 5 minutes 45 seconds. As shown in a recent study [PDPO15], notifications are more welcome when recipients are bored. However, our results show that while users might be willing to accept more notifications when idle, the time needed to attend to such notifications might be higher compared to the time needed to attend to a notification while a user is busy.

C. The Impact of Ongoing Task Complexity on Seen Time. To analyse the effect of ongoing task complexity, we first encode the reported task complexity, which was reported as a value on the Likert scale (*Strongly disagree*=1, *Somewhat disagree*=2, *Neutral*=3, *Somewhat agree*=4 and *Strongly agree*=5) to the question *"The task/activity I was doing when the notification arrived was complex"*. The Spearman's rank correlation coefficient is computed to evaluate the relationship between the complexity of an ongoing task and the seen time of a notification. The results show that there is a weak, negative correlation between the two variables, $\rho = -0.183$, $p = 0.005$. Thus, the increase in the seen time of notifications is correlated with the decrease in rating of ongoing task complexity. A possible explanation of this correlation is that users become more alert while

performing a complex task and, thus, quickly perceive the interruptions. On the other hand, when the users are not performing any complex task, they become less attentive to the interruptions.

Finally, we found that factors such as the completion level of the ongoing task and the sender type do not have a statistically significant effect on the seen time of notifications.

What Factors Influence the Decision Time?

We analyse the effect of the type, complexity and completion level of the ongoing task, and the sender type on the time a user takes to decide how to react to a notification. We find that neither of these factors have a statistically significant effect on the decision time of notifications with the exception of the sender.

A Kruskal-Wallis analysis of the decision time was carried out for each sender type. The results show that the sender type has an impact on the seen time of notifications with $\chi^2(10, 212) = 28.75$, $p < 0.001$. A Tukey post-hoc test (by setting α equal to 0.05) revealed that out of the 11 sender types (shown in Table 4.2), notifications from partner lead to the fastest decision time (mean DT is 3.31 seconds with the standard deviation equal to 1.71 seconds), followed by immediate family members with an average decision time of 4.89 seconds (with the standard deviation equal to 1.88 seconds). On the other hand, notifications from extended family members and service providers have the longest decision time, 11.93 and 8.14 seconds respectively (with the standard deviations equal to 3.12 and 3.31 seconds). There was no statistically significant difference in the decision time of the notifications from other senders. These results demonstrate that notifications are quickly handled when they are sent by the close relatives of the user. In other cases users take more time in reading the content before deciding how to handle it. We hypothesise that this behaviour stems from the content of notifications from close friends or family members, which might be more predictable, and a part of a daily routine (e.g., "pick kids from school"). *On the other hand, the users have to spend more time on the notifications from less frequently contacted sources, as the content may be less familiar to them.*

The Role of Notification Presentation

In our dataset, 2953 (out of 10372) notifications were received when the user was engaged with the phone. Out of these 2953 notifications, 860 are so-called "low-priority" while 2093 are "high-priority" notifications [And16b]. Here, a high-priority notification is a foreground notification that gets in the way of the user's ongoing activity; the user cannot perform any action to get it out of the way without clicking or dismissing it (e.g., Viber messages). A low-priority one simply appears on the notification bar without getting in the way of the user's ongoing activity (e.g., Gmail notifications).

We investigate the effect of the notification presentation on the response time (i.e., the sum of seen time and decision time) of a notification. The result of a two sample t-test shows that there is a statistically significant effect of notification priority on the response time, $t(2951) = 17.694$, $p < 0.001$. The mean response time for high-priority notifications is 11.94 seconds (with the standard deviation equal to 2.25 seconds) versus 25.91 seconds (with the standard deviation equal to 6.87 seconds) for low-priority notifications, which indicates that high-priority notifications get quicker response than low-priority notifications.

3.2.4 Why a Notification Becomes Disruptive

In this section we investigate the effect of different factors on the perceived disruption. Since the perceived disruption was measured with a 5-point Likert scale, we encode the responses as: *Strongly disagree*=1, *Somewhat disagree*=2, *Neutral*=3, *Somewhat agree*=4 and *Strongly agree*=5.

The Role of Ongoing Task Complexity

We now investigate whether the complexity of an ongoing task is associated with the perceived disruption reported by the users. A Kendall's Tau correlation coefficient was

The key findings of this section are:

- Perceived disruption increases as the complexity of an ongoing task increases.
- Notifications are perceived as most disruptive if they arrive when the user is in the middle of or finishing a task, and least disruptive if the user is idle or starting a new task.
- Messages from subordinates and system messages (where the sender is not a person) are considered as most disruptive. Whereas, extended family members are considered as the least disruptive.

Box 3.2: Key findings of the analysis for understanding why notifications are perceived as disruptive.

computed to assess the relationship between the ongoing task complexity and perceived disruption. We found a strong, positive correlation between the two variables, $R\tau = 0.477$, $p < 0.001$. This demonstrates that the users are likely to feel more disrupted by a notification that arrives when they are engaged in an intricate task and less disrupted when they are performing a simple task. Similarly, in our preliminary analysis [PMM15] we have found that when users are engaged in complex tasks they also express more of a negative sentiment towards interruptions.

The Role of Ongoing Task Completion Level

A Kruskal-Wallis analysis of the reported disruption was carried out for each class of task completion level (starting, in the middle, finishing and not doing anything). The results show that the completion level of an ongoing task has a significant impact on the disruption perceived by the users from the notifications, $\chi^2(3, 451) = 54.38$, $p < 0.001$. A Tukey post-hoc test (by setting α equal to 0.05) reveals that the perceived disruption is the highest when the user is currently involved in a task. The perceived disruption is the lowest when the user is starting a task or when they are idle but there is no statistically significant difference between these groups. These results demonstrate that when the user is deeply engaged in a task the perceived disruption is very high not only from the

desktop notification, as discussed for example in [CCH00a, MN86], but also from the mobile notifications.

The Role of Sender

We performed a Kruskal-Wallis analysis of the reported disruption for each type of sender (see Table 4.2). According to the results, $\chi^2(10, 444) = 36.27$, $p < 0.001$, the type of sender has a significant impact on the disruption perceived by the users from the notifications. A Tukey post-hoc test (by setting α equal to 0.05) reveals that the perceived disruption is highest when the sender is not a person or is a subordinate at work (no statistically significant difference between these two groups) and the lowest when the sender is an extended family member. Moreover, colleagues and service providers are the second most disruptive sender groups. There is no significant difference between the other groups. Previous studies showed that users express a negative *sentiment* towards messages not coming from their family and friends [FYB+10], and that the more "distant" the sender is, the less likely it is that a notification will be *clicked on* [MMHP15]. Results from our study complement this with the finding that the perceived *disruption* varies with the sender of a notification.

The Role of Ongoing Task Type

A Kruskal-Wallis analysis of the reported disruption is carried out for each type of ongoing task (see Table 4.2). The results show that the type of task that the user is engaged with (on the arrival of a notification) has a significant impact on the disruption the user perceives when the notification arrives, $\chi^2(5, 380) = 56.57$, $p < 0.001$. A Tukey post-hoc test (by setting α equal to 0.05) revealed that the perceived disruption is the highest when the user is working and the lowest while the user is idle. After work, traveling and then leisure are the tasks where the users perceive the highest level of disruption. When the users are not idle, they perceive least disruption while communicating and doing a personal or maintenance task. Since the communication can involve notifications

themselves (e.g., two mobile users exchanging WhatsApp messages) the above result is not surprising. As shown in a recent study [PDPO15], users are receptive to information when they are bored. Our results are in line with these findings in showing that perceived disruption is lowest when the user is idle.

3.2.5 Understanding the Acceptance of Notifications

In this section we investigate the factors that make the users accept (click) or dismiss a notification.

The key findings of this section are:

- Likelihood of the acceptance of a notification decreases with the increase in the perceived disruption.
- Disruptive notifications are accepted when they contain useful information.

Box 3.3: Key findings of the analysis for understanding the acceptance of notifications.

Procedure

Through the questionnaires, we asked the users the reason for clicking/dismissing a notification (see Table 4.2). If a notification (linked with the questionnaire) is clicked by the user, we ask them to select all factors that made them decide to click the notification, otherwise, we ask them to select the factors that made them decide to dismiss the notification. We provide a predefined list of seven options for clicking (see Table 3.3) and six options for dismissing (see Table 3.4) the notification. In addition, there is a box for open-ended answers in case users do not find an appropriate answer in the provided list.

In Table 3.3 and Table 3.4 we calculate the percentage of times each factor was reported as a reason for clicking and dismissing the notifications. Since users may select more than one option, the total count percentage in the table adds up to more than 100%. According

Option	Count (%)
Sender is important	31.546
The content is important	27.129
The content is urgent	14.511
The content is useful	31.546
I was waiting for this notification	15.773
The action demanded by the sender does not require a lot of effort	20.189
At this moment, I was free	37.224

Table 3.3: User response about why they accept (click) notifications. Note that users were allowed to select more than one option.

Option	Count (%)
Sender is not important	19.565
The content is not important	40.580
The content is not urgent	43.478
The content is not useful	38.406
The action demanded by the sender does require a lot of effort	3.623
At this moment, I was busy	19.565

Table 3.4: User response about why they dismiss notifications. Note that users were allowed to select more than one option.

to these responses, the users mostly accept notifications when they are free, but also the importance of the sender and the usefulness of the content make them accept a notification. Similarly, users avoid attending to notifications that do not contain important, urgent or useful content. These responses demonstrate that the value of the content is used when deciding whether to accept or dismiss a notification. Moreover, the users very rarely state that they were busy and thus had to dismiss a notification. This could indicate that the users give precedence to a notification over the primary task, but only if the content is valuable.

Disruptive Notifications are Likely to be Dismissed

We examine the impact of the disruption caused by the notifications on their likelihood of being accepted. In order to quantify this, we encoded the response for perceived disruption with the following values: *Strongly disagree*=1, *Somewhat disagree*=2, *Neutral*=3, *Somewhat agree*=4 and *Strongly agree*=5. In order to detect the acceptance of a no-

tification, we check whether it was clicked by the user. In case it was dismissed, we cross-validate the user's response for the question *How did you handle the notification?* If the user responded that *I decided to dismiss it because it didn't require any further action*, we mark this notification as accepted. Finally, we use 0 to indicate that the notification is dismissed and 1 for an accepted notification.

We fit a logistic regression model to estimate the effect of perceived disruption on the likelihood of the acceptance of notifications. The model was statistically significant $X^2(1) = 48.3$, $p < 0.001$. The results indicate the likelihood of the acceptance of a notification decreases by 0.581 times (95% confidence interval limits for the slope were [0.497, 0.675]) for a unit increase in the perceived disruption (based on 5-point Likert scale).

However, the coefficient of determination for the fitted model is not high ($R^2 = 0.1434$), which implies that the predictor (i.e., perceived disruption) does not explain much of the variation in the dependent variable (i.e., acceptance of notification). In other words, the acceptance of notifications could not be accurately predicted by the perceived disruption. This can be due to the fact that not only the disruption perceived by the user but also other factors influence the user's decision to accept a notification.

Why are disruptive notifications accepted?

As discussed above, the disruption perceived by the user makes a notification more likely to be dismissed. Our dataset shows that 104 out of 474 notifications (with which the questionnaires were linked) were reported as disruptive. These are the notifications for which the user *somewhat* and *strongly* agreed that they perceived disruption from these notifications.

However, 54% of these disruptive notifications were accepted (clicked) by the users, regardless of the fact that they caused disruption. To investigate the reason for this, we checked users' responses about the factors that made them click these notifications. Table 3.5 shows the percentage of times each factor was reported by the users for accepting

Option	Count (%)
Sender is important	25.926
The content is important	33.333
The content is urgent	20.370
The content is useful	35.185
I was waiting for this notification	11.111
The action demanded by the sender does not require a lot of effort	16.667
At this moment, I was free	18.519

Table 3.5: User response about why they accept disruptive notifications. Note that users were allowed to select more than one option.

the disruptive notifications. As users were allowed to select more than one option, the sum of the percentages in the table adds up to more than 100. "Content is important" and "Content is useful" are the most dominant reasons provided by the users for clicking the disruptive notifications. This tells us that even the notifications containing important or useful content can cause disruption. We suspect that these notifications may contain valuable information, but they were not relevant at the moment of delivery. However, our study does not provide sufficient evidence to support this conclusion.

3.2.6 Limitations

Most limitations of the work presented in this paper stem from our decision to collect data in the wild, with the minimum amount of intervention from our users. For example, when it comes to the computation of the seen time of a notification, we can only detect if a user unlocked the phone and assume that all notifications were seen. We cannot detect the precise time when a user starts reading a summary of a message from the notification bar. Moreover, in case a notification arrives when the user is already engaged with the phone, we assume that the user has seen the notification. Further, since our users are not confined to a laboratory, we are limited to recording the self-reported level of disruption from a notification. In reality, the impact on the primary task might not be high even if the perceived disruption level is high. On the other hand, this self perception might be the most important factor that determines the user's long term sentiment towards

notifications. When it comes to our ESM sampling, despite being as light as possible (we ask only up to four ESM questionnaires per day from each user), it increases the number of notifications a user sees during the data collection period. The density of notifications negatively impacts the sentiment towards individual notifications [PM14b]. However, we believe that in our case the impact is equally distributed among notifications, and consequently, that the findings about the role of different factors still hold.

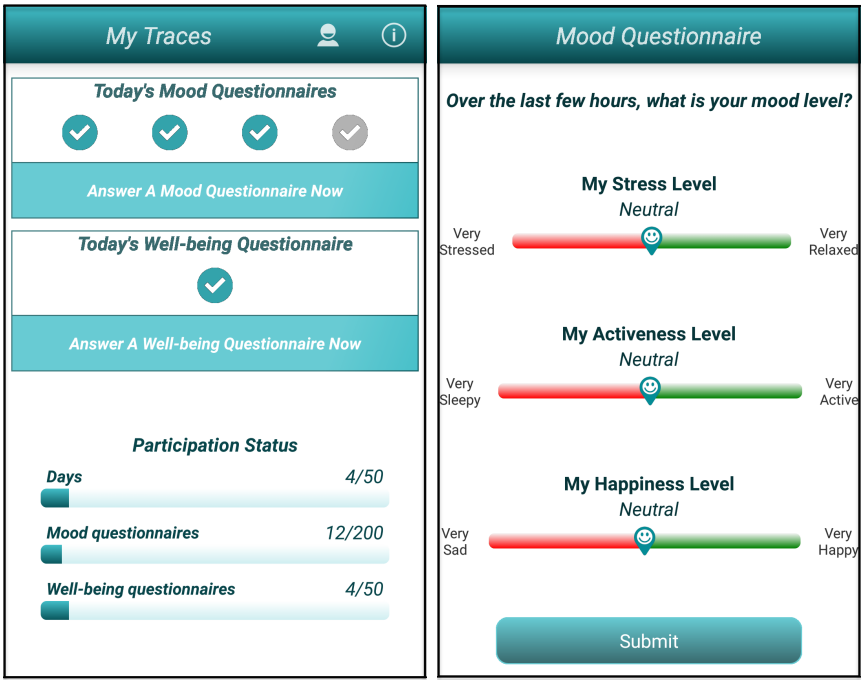
3.3 Understanding the Relationship between Emotional States and Notifications Response

In this section we present the results of second study that aims to understand the relationship between users' emotional states and their attentiveness and receptivity to mobile notifications.

3.3.1 Data Collection

In order to study the influence of emotional states on the user's mobile interaction behaviour, we conducted an *in-the-wild* study. We developed an Android app called MyTraces (shown in Figure 3.4) that runs in the background to unobtrusively and continuously collect the user's mobile phone interaction logs and their contextual information (as listed in Table 3.6).

The MyTraces application relies on Android's Notification Listener Service [And16a] to log interaction with notifications. It uses Google's Activity Recognition API [GAR16] to obtain the information about the user's physical activity (classified as walking, bicycling, commuting in vehicle or still). Moreover, the application samples GPS data in an adaptive sensing fashion as described in [CM15]. In order to cluster the GPS data we apply the clustering algorithm presented in [TM15]. 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 (defined as the time interval between 20:00 to



(a) (b)

Figure 3.4: MyTraces application: (a) main screen, (b) mood questionnaire.

08:00). We consider *work* place as the second most significant place (i.e., the place where users spend most of their time apart from home). All other places are labeled as *other*.

To acquire the data about a user's mood (activeness, happiness and stress level) throughout the day, we adopt an experience sampling method (ESM) [CL83]. As shown in Figure 3.4.b users can register their mood through a sliding bar. This bar uses a 5 point-based Likert scale where 1 indicates the lowest level and 5 the highest level. Every day a mood questionnaire is triggered four random times in every three hour time window between 8.00 am and 11.00 pm by using the phone's time (i.e., local time zones). We chose this time window so that the participants do not feel annoyed by being asked to respond to the surveys early in the morning and late at night. In case a questionnaire is dismissed or not responded to within 30 minutes from its arrival time, the application triggers another alert after 30 minutes.

Since the higher values of activeness and happiness levels indicate a positive emotion, we measured the stress level according to a negative scale that means lower value would indicate high level of stress. This way we make the scale of all emotional states consistent

Data Type	Features
Notification	Arrival time, seen time and removal time, alert type (sound, vibrate and flashing LED), user's response (click or dismiss), sender application name and notification title.
Context	Physical activity and location.
Phone Usage	Lock/unlock event, single click, long click, scrolls and usage time of all foreground applications (including home screen).

Table 3.6: Classes of data used for computing the phone interaction metrics and context-based features.

(i.e., the lower values refer to negative emotion and higher values to the positive emotion). Therefore, we reverse the scale by subtracting each response value from 6. So, if for example the response is 5 (i.e., very low stress), we subtract it from 6 to rescale it to 1. Thus, with the reversed scale the lower value will refer to lower stress and the higher value would indicate higher stress.

Recruitment of the Participants

The MyTraces application was published on Google Play Store and has been available to the general public for free since 4th January 2016. It was advertised through different channels: academic mailing lists, Twitter, Facebook and Reddit. In order to attract more participants for our study, we committed to give incentives to the participants for replying to the questionnaires for a minimum of 30 days. We committed to select (through a lottery) one winner of a Moto 360 Smartwatch and 20 winners of an Amazon voucher.

Ensuring Privacy Compliance

In order to ensure privacy compliance, the MyTraces application goes through a two-level user agreement to access the user's critical data. Firstly, the user has to give explicit permission as required by the Android operating system for capturing application usage, notifications and user's interaction with mobile phone (such as clicks, long-clicks and scrolls). Secondly, the application shows a list of information that is collected and asks for user's consent. Furthermore, the study was performed in accordance with our institution's

ethical research procedure and the consent form itself for the data collection was reviewed by our institution's Ethics Review Board.

3.3.2 Dataset

We consider the data collected from 4th January to 1st July 2016. In this period the application was installed by 104 users. However, many users did not actively respond to the mood questionnaires and some uninstalled the application after a few days. Therefore, we selected a subset of the data for the analysis by considering only the users who ran the application for at least 20 days and responded to at least 50% of the mood questionnaires in order to have a a sample sufficiently large to be statistically significant. Consequently, there are 28 users who satisfied these constraints. Note that we do not have information about the demographics of these participants because it was not asked during the study for privacy reasons. Our final dataset (i.e., the subset of active users) comprises 5118 responses to mood questionnaires, more than 9 million mobile interaction logs and 2 million context samples.

3.3.3 Definition of User Behaviour Metrics

In this section we introduce a series of metrics that are derived by quantifying users' emotional states, their interaction with mobile notifications and their context. These metrics represent the basis of our correlation and causality analysis that we will present in the following section.

Emotional States

We consider three aspects of emotional states that are captured during the day:

- **activeness level**: a state of being aroused and of physiological readiness to respond [Tha89, PM75];

- **happiness level**: a state of positiveness and joy that is derived from external and momentary pleasures [Sel04];
- **stress level**: a negative state of being under high mental pressure [Sel56].

The levels of these emotional states are computed on a 5 point-based Likert scale, where 1 indicates the lowest level and 5 the highest level.

Notification Metrics

We use the notification and phone usage data (described in Table 3.6) to compute six metrics that represent aggregate information about the user's receptivity and attentiveness to notifications. More formally, we define these metrics as:

- **Count**: Total number of notifications clicked.
- **Acceptance %**: Percentage of notifications clicked out of total arrived.
- **Percentage Handled (Other Device)**: Percentage of notifications that are not handled on phone out of total notifications arrived.
- **Average Seen Time (ST)**: Average of the seen time of all notifications. Seen time refers to the time from the notification arrival until the time the notification was seen by the user.
- **Average Decision Time (DT)**: Average of the decision time of all notifications. Decision time refers to the time from the moment a notification was seen until the time the notification was responded to by the user.
- **Average Response Time (RT)**: Average of the response time of all notifications. Response time refers to the time from the notification arrival until the time the notification was responded to by the user.

It is worth noting some additional details about the calculation of the metric "*Percentage Handled (Other Device)*", which represents the information about the user's engagement with other 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 of the phone if it was delivered on some other device and the user has already interacted with it on that device.

Context-based Metrics

We use the context data to compute two metrics:

- the time for which different activities are performed by the user.
- the time spent at different places by the user.

We compute both metrics on an hourly basis for each day. It is worth noting that these metrics are used as confounding variables for the causality analysis as discussed in the next section.

3.3.4 Methodology

Correlation Analysis

In this section we describe the methodology that we followed in order to study the relationships between emotional states (i.e., activeness, happiness and stress) and users' interactions with notifications. In order to quantify this association, we compute the individual-based Kendall's rank correlation coefficients. We consider the absolute values of these coefficients because we are interested in the strength of the relationships between the variables. We then compute the average of these coefficient values. We rely on Fisher's method [FY38] for combining the p-values of individual-based correlation analyses.

The correlation analysis is performed between the emotional state at the current hour (i.e., hour of the day in which the information about the user's emotion is acquired) and

the values of the examined notification metric for three different time intervals: *preceding, current* and *next hour*. Consequently, the final number of data samples in our analysis is equal to the total number of emotional state reports provided by the participants.

The correlation results are presented as a correlation matrix plot. In this matrix the y-axis indicates the notification metrics and x-axis indicates the type of emotions that are correlated with the metric computed for the specific hour. Here, the *hour* is represented by the numeric value on the x-axis labels. For instance, in Figure 3.5 the box in the first row (*Acceptance Percentage*) and the first column (*-1 Activeness*) presents the coefficient for the correlation of the activeness level with the acceptance percentage of notifications, computed by using data related to the $current - 1$ hour. Here, the current hour refers to the hour in which a user reported their emotional state. We set the significance level α for the correlation results to 0.05 and non-significant correlation coefficients are indicated by the white boxes in the correlation plots.

Causality Analysis

Correlation analysis reveals the relationship between emotional state and the notification metrics. However, a pure correlation between two variables does not necessarily imply the existence of a causal influence as the values of both the examined variables may be associated with other factors, i.e., they can be "explained" by other factors. For instance, while at work users might be less receptive to notifications and they might also feel stressed at work. In this case, a correlation between users' receptivity to notifications and their stress level might be observed. However, the values of both variables are strongly influenced by the user's location (i.e., work). Therefore, we perform the causality analysis between the variables that are significantly correlated.

In this study, we apply the causality analysis framework (as described in [TM15]) in order to analyse the impact of participants' emotional state on their interaction with notifications. In this case, the treatment variable is one of three emotional state-based metrics and the outcome variable is one of the notification metrics.

Since a causal link cannot exist between two uncorrelated variables and the treatment needs to precede temporally the outcome variable, we conduct this analysis only on variables for which a statistically significant correlation between emotional state and the previous hour's phone interaction metric has been observed.

Moreover, a separate causality study is conducted for each pair of variables. Additionally, we should stress that our study controls only for observed confounding variables (discussed earlier in Section 3.3.3) and could be biased in the case of missing confounders. Although several unobserved factors could influence the user's emotional state, bias could be induced only by those factors that also influence user's interaction with notifications. By including in our study a large number of metrics and by controlling also for the previous values of emotional state and notification metrics, we minimise this bias. Moreover, it is worth noting that the causal analysis performed in this study enables us to detect the potential causal effect in the absence of other confounders and in any case it provides evidence of strong dependency between variables.

3.3.5 Results

This section presents the results of the correlation and causality analyses for the reported emotional states and notification metrics. Figure 3.5 shows the correlation coefficients that are computed to assess the relationship between emotional states (activeness, happiness and stress) and six notification metrics. The results show that the activeness level moderately correlates with the average seen time (that we indicate by ST) and decision time (that we indicate by DT) of notifications that arrive in the *next hour*. This indicates that the users' awareness and pace for reacting to notifications is linked with their activeness level. This is in a sense expected, since a user who is less energetic might delay their response to notifications. Moreover, we also observe a moderate association between stress level and the average RT of notifications that arrive in the *next hour*. We believe that this correlation exists because users become more alert while performing a complex and stressful tasks.

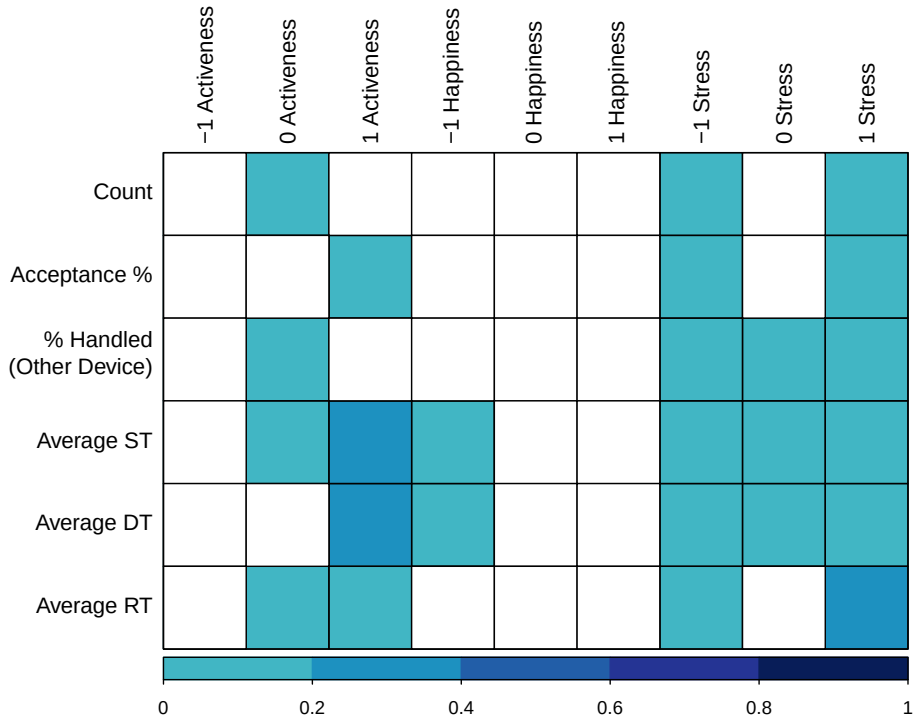


Figure 3.5: Results for correlation between emotional states and notification metrics.

	Average Treatment Effect		
Metric	Activeness	Happiness	Stress
Average ST	-22.18	NP	NP
Average DT	-4.09	NP	NP
Average RT	NP	NP	-71.21**
** refers to the p-values <0.005.			
NP refers to not present.			

Table 3.7: Results for the causal effect of emotional states on notification metrics.

In order to investigate if there is any causal link here, we perform causality analysis to quantify the impact of activeness on the average ST and DT of notifications, and the impact of stress on the average RT. Table 3.7 presents the average treatment effect that describes the strength of the causal relationship (as discussed in [TM15]). The results indicate that the activeness level has no statistically significant impact on the average ST and DT. This means that there is some other variable (such as time or location) than drives both activeness as well as the average ST and DT. For instance, people's activeness might vary with their location and so does the ST and DT of notifications.

However, we observe that there is a statistically significant and negative causal impact of stress on the average RT (i.e., response time for notifications). The negative value indicates that the average response time to notifications is lower for participants with higher stress score (i.e., more stressed). This suggests that people become more attentive to notifications when they are stressed.

Quite interestingly, in the first study presented in this chapter [MPV+16] we have also found that users' attentiveness increases as the complexity of an ongoing task increases. Results from our study confirm this also highlighting the presence of a causal link. Note that we do not assess causality between other variables as they do not show significant and moderate association (i.e., the correlation coefficient is greater than 0.2). Indeed, a necessary condition for causality between two variables is the presence of correlation in the first place. Therefore, we indicate NP (Not Present) for these relationships in the table.

On the other hand, we also do not see any notification metric for the *preceding hour* that has a significant and moderate relationship with an emotional state. Therefore, no causality analysis is performed for assessing the impact of these metrics on emotional states.

3.3.6 Limitations

In this work we have studied the association between users' emotional states and their interaction with notifications. We have initially conducted a correlation study in order to detect links between user mood and phone interaction. Then, we have attempted to understand the causal impact of users' mood on their interaction with notifications and vice versa.

Our study utilises raw sensor data in order to extract high-level information such as location labels and activity, and, consequently, it is subject to limitations and inaccuracies of the inference method used to extract high-level information. Moreover, self-reported emotion states may be in themselves inaccurate, given the known problems related to biased

self-representation in questionnaires [CL14], or not answered very frequently [MVPM15]. Considering also that some users may not be willing to answer any questionnaires when they are very stressed, unhappy or sleepy, our study may fail to capture such extreme cases. This is a common limitation of all smartphone-based studies and, for this reason, we believe it is fundamental to perform additional experiments to reproduce and re-validate these results in different settings in the future.

Furthermore, our causality study is based on observational data. Causal inference on observational data could be biased in the case variables that have a direct influence on both the treatment and the outcome variables are not included in the study or are not sufficiently controlled [SCC02]. In our case, we control several factors that influence both emotional state and the specific notification interaction metric taken into consideration, such as other notification interaction features, location, and activity (including both current and past values).

3.4 Summary

This chapter presented two studies for investigating the factors influencing users' attentiveness and receptivity to mobile notifications.

The first study focussed on mobile interruptibility, specifically on the identification of factors that make an interruption disruptive and the impact on the response time to a notification. The contributions of this study are twofold. First, we have confirmed the validity of some past desktop interruptibility studies [CCH00b, CCH00a, CCH01] to show that ongoing task's complexity and completion level influences the perceived interruption in a mobile setting. Second, for the first time, we have investigated the role of notification presentation, sender-recipient relationship and emotional states for modelling interruptibility. More specifically, the key contributions are listed in Boxes 3.1, 3.2 and 3.3. Through a mixed method of automated smartphone logging and ESM sampling we have

obtained a dataset of *in-the-wild* notifications and ESM reports on notification perception from 20 users. We have analysed the data to show that the response time of a notification in the mobile environment is not only influenced by an ongoing task's type, completion level and task complexity, but also by the notification's alert modality, presentation and sender-recipient relationship. Our results have shown that the presentation of a notification and its alert type, as well as the type, completion level and complexity of a task with which the user is engaged, all impact the seen time. Moreover, the relationship with the sender influences the user's decision on accepting a notification or not. Finally, the data also reveals how the sentiment (i.e., perceived disruption) towards a notification varies with the type, completion level and complexity of an ongoing task and the recipient's relationship with the sender.

In the second study we have performed a causality analysis between users' emotional states and their interaction with notifications. We collected 5118 responses to questionnaires for logging users' emotional states (activeness, happiness and stress) from 28 users over a period of 20 days. First of all, using a non-parametric correlation test (Kendall's Rank) we have shown that the users' activeness level has a significant association with the seen and decision time of notifications that arrive in the *next hour*. Then, we have conducted an in-depth causality analysis considering a variety of contextual variables as confounders to show that in stressful situations people become more attentive that results in the reduction of notification response time.

CHAPTER 4

PREDICTING OPPORTUNE MOMENTS TO DELIVER NOTIFICATIONS

4.1 Overview

Users' interaction with mobile notifications is indeed extremely complex and depends on numerous aspects. Fortunately, some of the aspects can be captured, and above all, *quantified*, by means of the embedded sensors with which modern mobile phones are equipped. These sensors allow a mobile application to collect information about users' day-to-day activities [CMT+08, MPM14], preferences [XLL+13], and the surrounding environment [LPL+09]. Past studies have investigated the use of some of the sensed information to infer *opportune moments* for interruptions, i.e., moments when a user quickly and/or favourably reacts to a notification [FGB11, PM14b]. More specifically, some important contextual factors that have been used to infer interruptibility include transitions [HI05], engagement with a mobile device [FGB11], and, more generally, time of day, location and activity [PM14b].

However, until now, the focus has been on the *context* in which a notification has been received and not on the actual *content* of the notification. After all, not all the notifications are disruptive [ML02]. It is *the relevance of the interruption content in the recipient's current context* that partly defines the disruptiveness of an interruption. For

example, a chat notification from a friend can be extremely disruptive if delivered during a meeting. But, an email notification from a project collaborator might be acceptable to the user, and in some cases, considered very useful in the same context.

In this work we discuss how content and context can be used *together* in order to design intelligent non-disruptive notification mechanisms. More specifically, we investigate how users behave when they receive specific types of content through mobile notifications arriving at different times, and in different contexts. Unlike previous studies such as [PM14b], we do not restrict ourselves to context information provided by mobile sensors only. Instead, to the best of our knowledge, for the first time we also take into account the type of information delivered and the social relationship between the information sender and receiver. It is worth noting that our work is predicated on the hypothesis that the acceptance of a mobile notification depends on *what*, i.e., the type and the origin of the information contained in a notification and *where*, i.e., the user's context in which the notification is delivered. We will consider a notification as accepted if it is handled (i.e., clicked to launch the respective application) by the user within a certain time from its arrival.

4.1.1 Key Contributions

The findings of our analysis can be summarised as follows:

- user's acceptance of a notification depends on the content and the originator of the notification and, not surprisingly, the time needed to respond to a mobile notification varies according to the user's physical activity level.
- notification content, together with the sensed context, can serve as a basis for the design of machine learning classifiers that infer a user's response to a notification.
- automated inference of opportune moments to interrupt, as with the above classifiers, outperforms subjective rules with which our subjects described their interruptibility.
- the inference of a user's interruptibility can be performed locally, in an online learn-

ing fashion, achieving a stable state (i.e., more than 60% precision) after nine days of training.

We conclude that inferring the right moment for interruption should be done in a holistic manner, jointly taking into account the context of the recipient and the notification content. We show that a prediction model trained on a user's personal data performs far better than a generic prediction model trained on multiple users' data. Finally, we point out the importance of an automated approach, as humans remain inefficient at formalising and communicating their preferences for mobile notifications in terms of predefined rules.

4.2 Opportune Moments to Deliver a Notification

4.2.1 What Defines an Opportune Moment?

As we discussed in Chapter 2, in [Cla96] Clark suggests that a user can respond to an interruption in four possible ways:

1. handle it immediately;
2. acknowledge it and agree to handle it later;
3. decline it (explicitly refusing to handle it);
4. withdraw it (implicitly refusing to handle it).

The first two categories of responses suggest that the interruption content is acceptable to the user. However, the second type of response indicates that *the interruption content is not relevant at the moment of its delivery* and, thus, it should have arrived after a certain time delay in order to be handled immediately. On the other hand, the responses with a decline and withdraw might indicate that the interruption is not at all acceptable to the users.

We construct our hypothesis based on these possible responses of a user to an interruption. We hypothesise that a moment is opportune to deliver a notification only if the user handles the notification immediately. This does not mean that an intelligent inter-

ruptibility management system should just try to reduce the response time, it should also aim to increase the acceptance rate of notifications.

4.2.2 How to Predict an Opportune Moment?

Sahami et al. [SSHD+14] demonstrated that notifications triggered by applications from various categories are given different importance by users. At the same time, Pielot et al. [PdOKO14] claimed that users' responses are influenced by social pressure. This suggests that users' decisions about how to respond to a notification are made after looking at the notification title which gives a clue about the information contained in it. On the other hand, Grandhi et al. [GJ09] suggested that by just relying on sensor based knowledge, interruption management systems often fail to infer whether the interruptions are disruptive or not because they do not take into account the sender of an interruption and the information that is being sent.

Thus, better predictions can be made to infer an opportune moment to deliver a notification by considering both notification content and the context in which it is delivered. To better understand the definition of opportune moments, let us consider a scenario in which a person at their workplace receives a notification regarding the arrival of an email from a colleague working on the same project and, separately, a new comment on one of their social networking posts. Given the circumstances, the user accepts the email notification leaving the social networking notification unattended. Later, on the way home, the user clicks on the social networking notification to read the new comment.

In this scenario, since only one notification was read while the other was unattended, it is impossible to decide whether the sending moment is an opportune moment or not by using only the context of the user. Both notifications arrived at the same moment but were accepted in different contexts. It is the notification content that enables the recipient to decide whether to accept it or not in the current context. Thus, the notification content and the user's context together play an important role in identifying an opportune moment to deliver a notification. However, due to privacy reasons, it might not be feasible

Feature	Description
Arrival time	Time at which a notification arrives in the notification bar.
Removal time	Time at which a notification is removed from the notification bar.
Response time	Difference between arrival and removal time.
Notification response	Whether the notification was clicked or not (boolean).
Sender application	Name and package of an application that triggered a notification.
Notification title	Title of a notification displayed in the notification bar.
Alert type	Signals used to alert the user for a notification: sound, vibrate, and LED.
Physical activity	Current activity of a user.
Location	Current location of a user.
Surrounding sound	Whether the user is in a silent environment or not (boolean).
WiFi connectivity	Whether the phone is connected to WiFi or not (boolean).
Proximity	Whether the user was proximate to the phone in the last one minute or not (boolean).
Phone's status	Whether the phone was in use in the last one minute or not (boolean).
Ringer mode	Current ringer mode: sound, vibrate and LED.

Table 4.1: Description of features from the NotifyMe dataset.

to exploit the content of notifications because they might contain extremely sensitive information.

Therefore, we propose to use the notification title that contains more high-level and abstract, but, at the same time, useful information about the category of information contained in the message itself. More specifically, the title can be used to classify the category of a notification. In addition, we capture a series of attributes about the user's physical context as well as the phone settings. These content and context data are then used to build a prediction model to predict the acceptance of a notification. The study was done in accordance with our institution's ethical research procedures, and all the participants were volunteers who provided their consent to data collection. The consent form itself for the data collection application was reviewed by the Ethics Board of our institution.

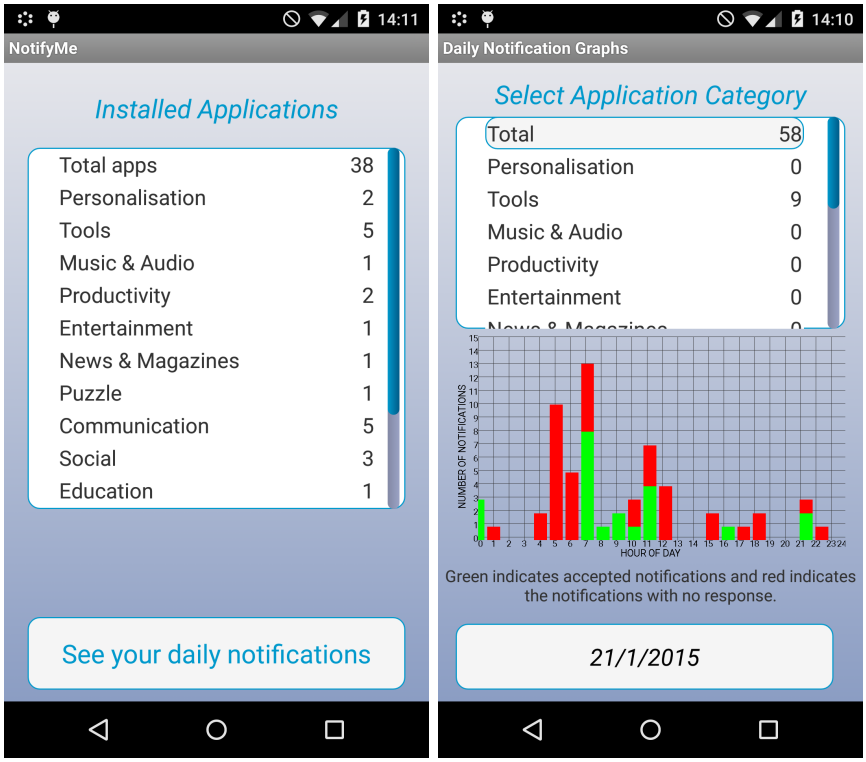


Figure 4.1: NotifyMe application screenshots.

4.3 Data Collection

In order to study the influence of context and content on users' responses, we collected a dataset of *in-the-wild* notifications. We developed an Android app called NotifyMe (shown in Figure 4.1) that runs in the background to unobtrusively monitor notifications and the context in which they are posted. It relies on Android's Notification Listener Service [And16a] to trace notifications, and it uses Google's Activity Recognition API [GAR16] and ESSensorManager [LRMR13] to obtain the context information. Table 4.1 describes the features captured by NotifyMe. It is worth noting that *Alert Type* indicates the signals used by a notification to alert the user, whereas *Ringer Mode* refers to the phone's ringer mode settings.

To infer the user's response to a notification, NotifyMe checks whether the application that triggered the notification was launched after the removal time of that notification. Since most of the notifications are triggered by chat and email applications, users usually click on such notifications to read the full text and reply. Therefore, we assume that

Question	Options
How would you rate the notification content?	Likert scale rating between 1 and 5 (1 = very annoying and 5 = very interesting).
Where would you like to receive notifications with similar content?	Home, workplace, other, anywhere and I don't want.
When would you like to receive notifications with similar content?	Morning, afternoon, evening, night, anytime and never.
How are you feeling?	Happy, sad, bored and annoyed.
Are you busy?	Yes and no.
Where are you?	Home, workplace, public, other.

Table 4.2: Questions and their options from NotifyMe questionnaire.

users click on the notifications that arrive at opportune moments. We are aware that our approach has limitations, because some notifications, which do not require further action, might not be clicked rather just seen and dismissed by the user.

Additionally, every day, NotifyMe posts 12 notifications on the user's smartphone containing information that is randomly chosen from breaking news, weather updates, and Facebook likes. These notification are triggered randomly every hour between 8.00 am and 8.00 pm. In this way we generated a consistent set of notifications for our analysis over all the users and we enhanced the richness of the categories of information received by them. Furthermore, NotifyMe also collects subjective data from its users by triggering six questionnaires each day. A questionnaire consists of six multiple-choice questions. Each questionnaire is triggered at a random time in every two-hour time window between 8.00 am and 8.00 pm. When users are busy, NotifyMe allows them to decline the questionnaire notification by simply removing it from the notification bar; moreover, the application makes sure that it does not trigger another questionnaire for the next 30 minutes. The list of questions, along with the options included, are shown in Table 4.2.

4.3.1 Recruitment of the Participants

NotifyMe was published on Google Play Store and advertised at our University. It was installed by 35 participants without any monetary incentive. These participants come

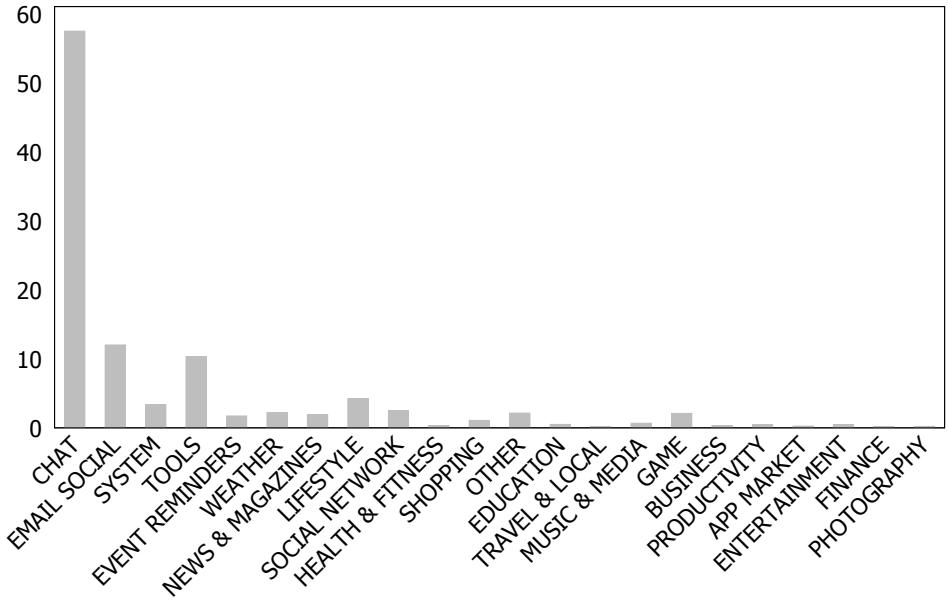


Figure 4.2: **Percentage of notifications for each category.**

from both sexes, with the age span between 21 and 31 years. As shown in Figure 4.1, NotifyMe allows users to check the number of installed applications for each category and displays a bar graph of their notification acceptance rate for each hour of a day. We believe that displaying this information has a minimal interference on users' actual behaviour in terms of notification acceptance, yet presents a valuable stimulus to make the users keep the app installed on their phones.

4.3.2 Ensuring Privacy Compliance

In order to allow the NotifyMe application to monitor notifications, users have to give explicit permission as required by the Android operating system. Moreover, the application also requires a detailed user consent about the information that is collected. This ensures that users are aware of the type of information captured by the application. Furthermore, we use notification content only to classify the category of information that a notification contains because storing and analysing such data can have severe privacy implications. We only store the notification's content category and discard the raw notification information.

4.4 Dataset

The data collection was carried out for a time period of 3 weeks from 35 users who installed the NotifyMe application. Overall, we collected more than 70000 notifications and around 4069 responses to questionnaires. Since, our objective is to predict the probability of a notification's acceptance by using a notification's content category and the user's context when it is delivered, we classify the collected notifications according to the type of information they contain.

In order to classify a notification, we use the approach of mapping the notifications to the categories of the applications from which they were triggered. The categories defined by the Google Play Store are too generic. Therefore, we manually categorise the applications from which the notifications were triggered into 23 categories (shown in Figure 4.2).

Figure 4.2 shows the percentage of notifications collected for each category. Around 70% of the data set comprises of notifications triggered by chat and email applications. Since these two notification categories dominate over the combined notification counts of all the other application categories, we could not rely solely on the approach to classify notifications based on the category of applications with which they were triggered. Thus, we split chat and email notifications in the following four sub-categories based on the sender's relationship with the recipient of a notification:

1. Work: sender works or studies with the recipient;
2. Social: sender has a social tie with the recipient;
3. Family: sender is the recipient's family member or relative;
4. Other: sender not related to the recipient with the above relations.

In practical applications, this information might be extracted from social network platforms and/or inserted directly by the users (see also a more detailed discussion in section 4.9). We believe that this methodology can be generalised: more fine-grained or different classification might also be possible.

We rely on the title of communication notifications in order to classify them in the above four classes. A communication notification's title comprises of the sender's name along with a short description about the newly available information (such as "A new message from Alice", and "Alice sent you a new message"), or only the sender's name (such as "alice@gmail.com" or "Alice"). However, a few communication applications do not disclose any information about the sender in the notification title but only includes the description about the newly available information (such as "1 new message") or the application's name itself (such as "Telegram"). Therefore, we use the notifications that contain the sender's information in their title and discard the rest of the communication notifications. We then generate a list of unique titles[1] from the communication notifications of each participant and give these lists to their owners. We ask the participants to label each notification title with the following classes: work, social, family, and other. Participants were asked to provide multiple labels in case they have multiple relational links with the sender of a notification. For example, a colleague can also be a friend of a user.

Finally, we map the user-derived labels for the titles to their corresponding notifications. Since, there were a few notification titles that were labeled with multiple classes: 1) work and social, and 2) social and family, we choose a category based on the location in which the notification arrived. For example, if a notification that is labeled as both social and work, arrives at their workplace we consider it as a notification containing work related information. We define a category probability list for each location from our data and look for a label that has the highest value at the location in which the notification was triggered. The category probability list for each location is defined as follows:

1. Workplace: work, social, family;
2. Home: social, family, work;
3. Other: family, social, work.

[1]In order to avoid the users labelling a title multiple times, we create this list containing notification titles without any duplication.

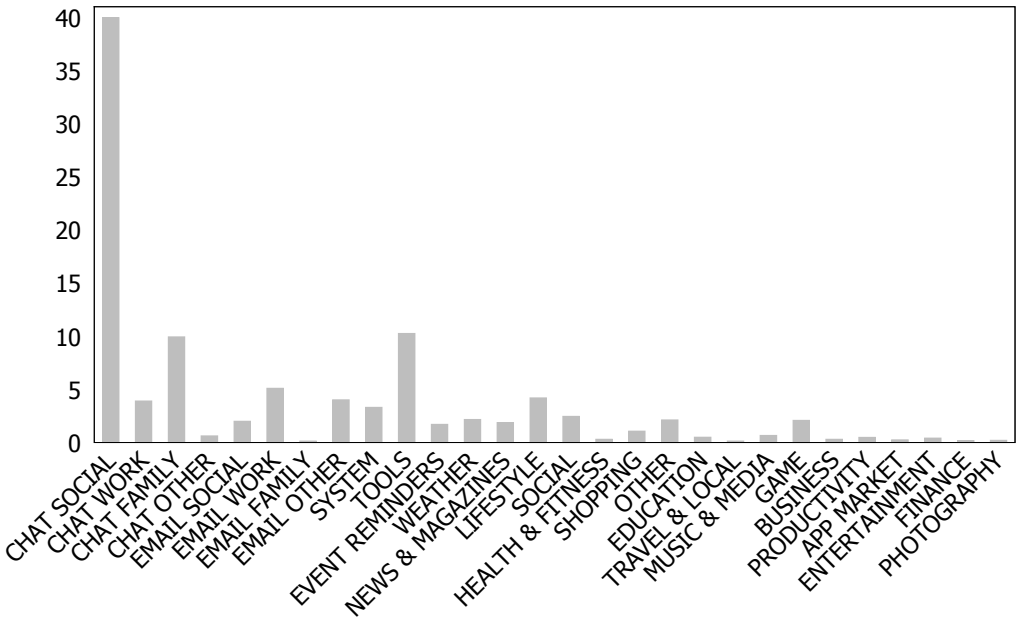


Figure 4.3: **Percentage of notifications for each category and sub-category. The sub-categories are derived by using the recipient's relationship with the sender.**

Note that in order to infer these three location classes we sample GPS when a questionnaire is answered. We assign the label for the current location provided by the users as an answer to a question "Where are you?" in the questionnaire, to the sampled geo-coordinates. Since the location labels provided by the users do not include any public location label, we exclude this location class. Additionally, we reverse geo-code the home address provided by the users at the time of registration and validate user-derived labels for these locations.

Out of 35 participants only 17 participants opted to label their communication notifications' title as discussed above. All of these participants continued to run the NotifyMe application and actively responded to the questionnaires for at least 15 days. Therefore, in order to perform notification category-based analysis, we use around 25000 notifications and 1450 questionnaire responses of the participants who opted to label the communication notifications' titles. Figure 4.3 shows the percentages of notifications that belong to each fine-grained category.

4.5 Understanding Interruptibility

In this section we provide evidence that the content and context play an important role in influencing the response time and acceptance of a notification.

4.5.1 Time Delay in Response to a Notification

As discussed earlier, an intelligent interruptibility management (IM) system should define a moment as opportune to deliver a notification only if the user handles the notification within a given time interval. Therefore, it becomes necessary to define a threshold for time delay in response to the notifications by the users. The notifications that are responded to after this threshold will be classified as delivered at an inopportune moment. This approach is based on the strategy employed by a previous study presented in [PdOKO14], which predicts whether a user will view a message within the assumed threshold time.

We analysed the complete dataset of notifications (containing around 70,000 notifications) to draw a picture of the general response time for all notifications. Note that a notification is considered responded to when it is removed from the notification bar.

As shown in Figure 4.4, more than 60% of the notifications were clicked within 10 minutes from the time of arrival. The density of notifications increases with a high rate up to 10 minutes and after 10 minutes the rate starts stabilising. This demonstrates that users handle most of the notifications by either clicking or dismissing them within this 10 minute period. Also, users do not interact with the notifications for a long time if they are not handled within 10 minutes. Overall, we conclude that in general the maximum response time taken by a user to handle notifications that arrive at opportune moments is 10 minutes. However, it might vary according to users' behaviour. Therefore, in this study we choose 10 minutes as the *threshold time delay* for responding to a notification. At the same time, the goal of this analysis is not strictly dependent on the choice of this specific value. In other words, the aim is to provide a general design methodology for intelligent content-driven notification systems.

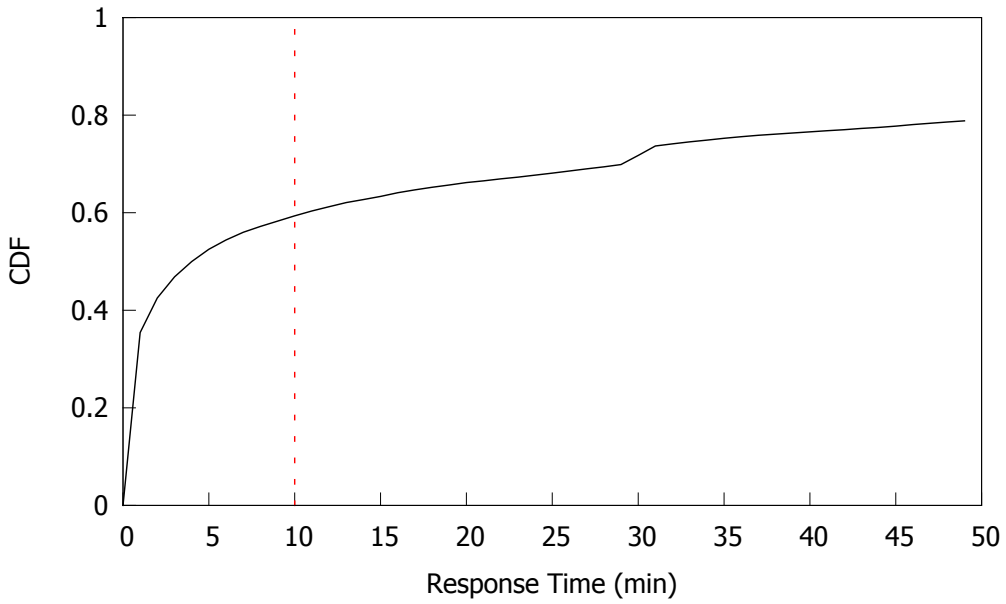


Figure 4.4: **Cumulative distribution function (CDF) of response time for notifications.**

It is also worth noting that there is a sharp increase in the number of responses at around 30 minutes after their arrival. This is because some applications kill their notifications that do not get any response from the users up to this threshold. Our dataset shows that more than 70% of the notifications which were removed from the notification bar between 29 and 31 minutes were triggered by Google Now.

4.5.2 Impact of Context on Response Time

In the collected dataset we observe that the notification response time can vary from a few seconds up to some hours. A notification delivered at an opportune moment might be responded to very quickly, but a notification can have a greater time delay if it arrives at an inopportune moment. We analyse the complete dataset of notifications to find the context modalities (i.e., the attributes of a user's context such as location, activity and others) that could indicate the response time for a notification.

We evaluate the response time of notifications with respect to three context modalities: location, activity, and background sound. We find that the average response time of a

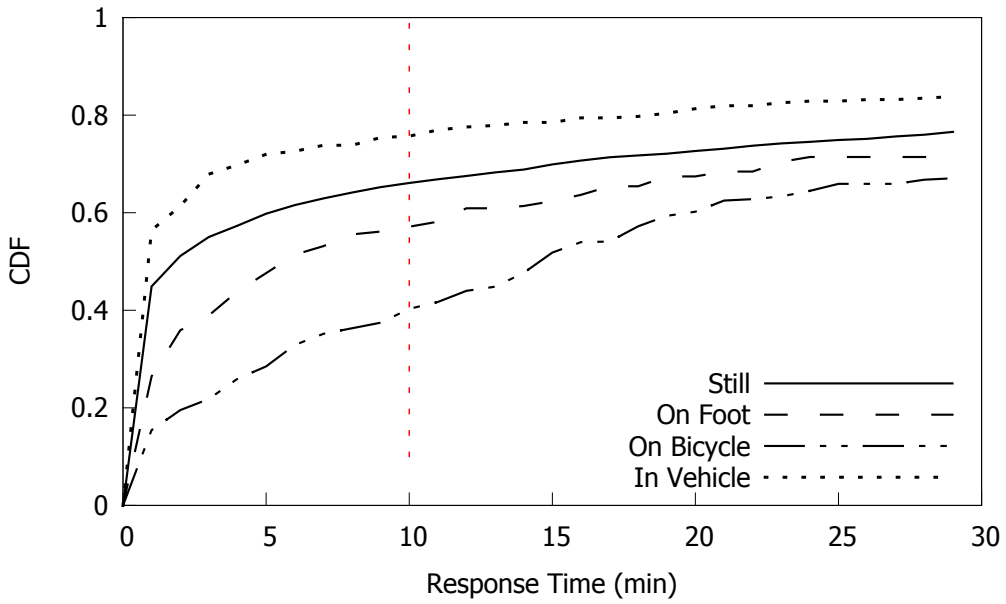


Figure 4.5: **CDF of response time for notifications received while performing different activities.**

notification does not vary with the user's location (i.e., home, workplace, and other) or the background sound (i.e., silent or speaking). However, the data shows that the activity of users does impact the response time of notifications.

We rely on Google's Activity Recognition library [GAR16] to classify a user's activity into four classes: 1. still – when the user is not moving; 2. on foot – when the user is either walking or running; 3. on bicycle – when the user is riding a bike; 4. in vehicle – when the user is travelling in a vehicle.

As shown in Figure 4.5, users are very quick to respond to notifications while they are traveling in a vehicle. Around 80% of the notifications that arrived while a user was in a vehicle were responded to within 10 minutes. At the same time, when users are still, walking, or running they tend to respond to almost around 60% of the notifications within 10 minutes. It is worth noting that the response rate gradually becomes flat after 10 minutes in all of these three cases. On the other hand, the activity of riding a bike is associated with inopportune moments to deliver a notification.

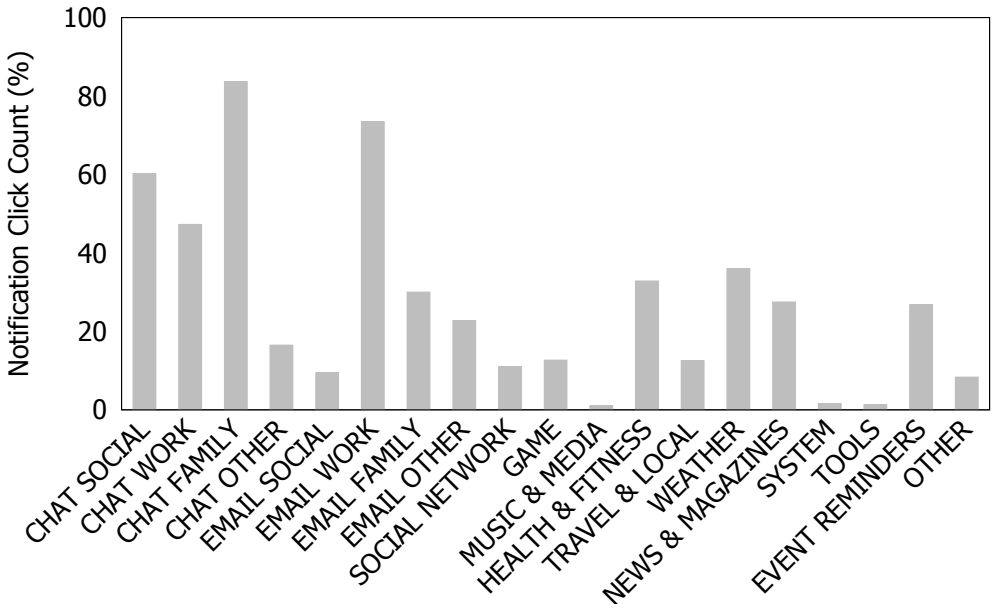


Figure 4.6: Click count percentages for the notifications of each category.

4.5.3 Impact of Content on Notification Acceptance

In this subsection we analyse the impact of the content on the acceptance of notifications. Since we are interested in the accepted notifications, we classify all the notifications with a response time greater than the threshold time (i.e., 10 minutes) as declined notifications.

In Figure 4.6 we evaluated the average percentage of the notifications that are accepted in a day for each notification category. The results demonstrate that the notifications from different categories have a varying acceptance rate. The family chat and work email notifications have the highest acceptance rate with around 81% and 77% of the notifications accepted within 10 minutes of arrival time, respectively. On the other hand, notifications of categories such as system, tools, and music and media, have the worst acceptance. Around 10% of the overall notifications in the dataset of labeled notifications belong to the tools category (see Figure 4.3) but around 99% of the time these notifications were declined by the users. This demonstrates that such notifications almost never provide useful information at the right time to the users and instead become a source of disruption.

It is worth noting that the event reminder notifications might have a low acceptance rate because 1) these notifications provide almost all the information in the notification

title; 2) these notifications enable the user to snooze, or delete the reminder through the notification itself without launching the application. For this reason, it becomes difficult to capture the response of a user for an event reminder notification.

4.6 Predicting The Right Time for a Notification

In this section we discuss the design of machine learning models for predicting for users' response (i.e., accept or dismiss) to a notification based on the type of information in it and the given context.

4.6.1 Data Setup

Notifications that are delivered at inopportune moments are usually handled with a potentially long delay. Therefore, we label all notifications which were accepted within the threshold response time (i.e., 10 minutes) as "accepted" and the rest as "declined".

4.6.2 Feature Ranking

To understand the importance of these features, we rank each of them based on the information gained by adding it for predicting the notification acceptance. We use the *InfoGainAttributeEval* method from WEKA [HFH+09] to derive the information gain (IG) each of the attributes brings to the overall classification of a notification (i.e., accepted or declined). Table 4.3 shows the average ranking of the features. The feature evaluation is based on a 10-fold cross validation. Our results show that the name of the application from which a notification is triggered and the notification category are the most important features.

Feature	Rank	Average IG
App Name	1	0.251
Notification category	2	0.247
Phone status	3	0.092
Location	4	0.081
Arrival hour	5	0.073
Ringer mode	6	0.056
User's activity	7	0.042
Priority	8	0.026
Alert type	9	0.024
Proximity	10	0.017
Surrounding sound	11	0.003
WiFi connectivity	12	0.001

Table 4.3: Ranking of features from the NotifyMe dataset.

4.6.3 Building Prediction Models

We build individual-based models for predicting notification acceptance by using three different algorithms: Naive Bayes, AdaBoost, and Random Forest. We use these three algorithms as they represent the state-of-the-art in machine learning for this type of problems [PdOKO14, PDPO15]. Moreover, we use three different machine learning algorithms to check whether the knowledge about the user's interruptibility can be obtained better by a specific type of algorithm.

We use two approaches for building prediction models: 1. Data-driven learning that relies on the automatically recorded previous interactions with notifications; 2. User-derived rules that rely on the user's own intuitions.

Data-driven learning

The data-driven learning approach relies on all the features about the notifications that are collected via the NotifyMe application. As discussed earlier, we derive the category of a notification by using the type of information in it and the recipient's relationship with the sender. However, in order to evaluate the value of using the information type and social circle, we build the prediction models in three ways:

1. without using information type and social circle;

2. using only information type;

3. using information type and social circle.

To build a prediction model in the first and third ways, we simply excluded and included the *"category"* feature respectively while training the algorithms. However, in order to train a predictor in the second way, we use modified the *"category"* feature that is derived by using only the information type. For example, work email was labeled as *email*, and social chat was labeled as *chat*.

User-derived Rules

The user-derived rule-based learning relies on users' responses to the questionnaires triggered by NotifyMe application. We collected 1456 questionnaire responses from the 17 users participating in our study. The analysis of these responses shows that users are not consistent in defining the rules for the delivery of notifications. Therefore, we used the following 3 features from each questionnaire response (see Table 4.2) to build a prediction model:

1. *notification category* for which the questionnaire was triggered;

2. *best location* where the user wants to receive notifications with similar content;

3. *best time* when the user wants to receive notifications with similar content.

The features in each questionnaire contributed to a rule for acceptance of a category of notifications. For example, a questionnaire response by a user containing "social chat" as content category, *"home and other"* as the best location, and "morning" as the best time defines that all social chat notifications that are delivered in the morning when the user is at home will be clicked. Otherwise, the social chat notifications will not be clicked by the user.

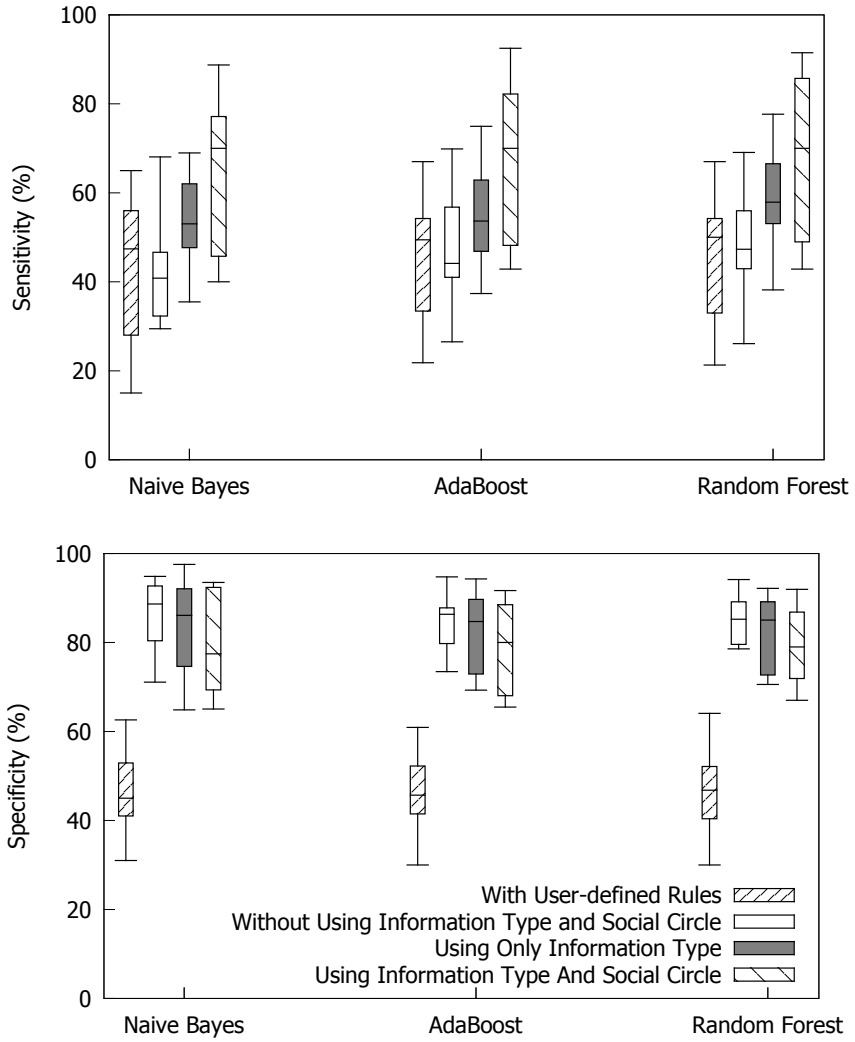


Figure 4.7: Prediction results of the predictors trained by using 3 different set of features for data-driven learning and user-derived rules.

It is worth noting that we construct user-derived models with a limited features obtained through the questionnaires. We asked limited questions about a notification in order to not to make the questionnaire time consuming and annoying for them. Therefore, these models do not indicate the best models that can be defined by users.

4.6.4 Evaluating the Predictors

We evaluate the data-driven prediction models by using the k-fold cross validation approach with the value of k equal to 10. To evaluate the prediction models for user-define

rules, we use all notifications of the respective users. Figure 4.7 shows the sensitivity and specificity of all the predictors. The sensitivity refers to the proportion of actual positives which are correctly identified (i.e., number of notifications that are correctly predicted as accepted divided by the number of notifications that are actually accepted). The specificity refers to the proportion of actual negatives that are correctly identified (i.e., number of notifications that are correctly predicted as declined divided by the number of notifications that are actually declined).

Our results demonstrate that there is no significant difference in the performance of the three prediction algorithms. The user-derived rules show worse performance compared to the data-driven learning approach. The user-derived rule-based predictors only achieve a sensitivity equal to 55% and a specificity equal to 45%. A possible explanation is that the user preferences change with time. For example, users might specify that they want to receive social chats only in the morning, but on the next day they might specify another time period for the same category of notifications. Moreover, users can define very abstract rules compared to the complex rules created with the data-driven approach by using numerous other features.

As we trained the data-driven predictors with different features, the results show that the predictor trained with "information type and social circle" indeed outperforms all the other predictors. The predictor trained with "information type and social circle" achieves a sensitivity equal to 70% and specificity up to 80%. It attains high sensitivity by losing some specificity compared to the other two data-driven predictors because of the trade-off between sensitivity and specificity. It is worth noting that the user preferences change with time. Therefore, by including the noisy features from the user-derived model, the accuracy of the data-driven predictor will definitely go down.

However, the aim of an interruptibility management system is not only to reduce the disruptive notifications (i.e., high specificity), but also to provide the notifications that are required by the user. This suggests that the use of information type and the recipient's relationship with the sender can boost the performance for predicting interruptibility. It

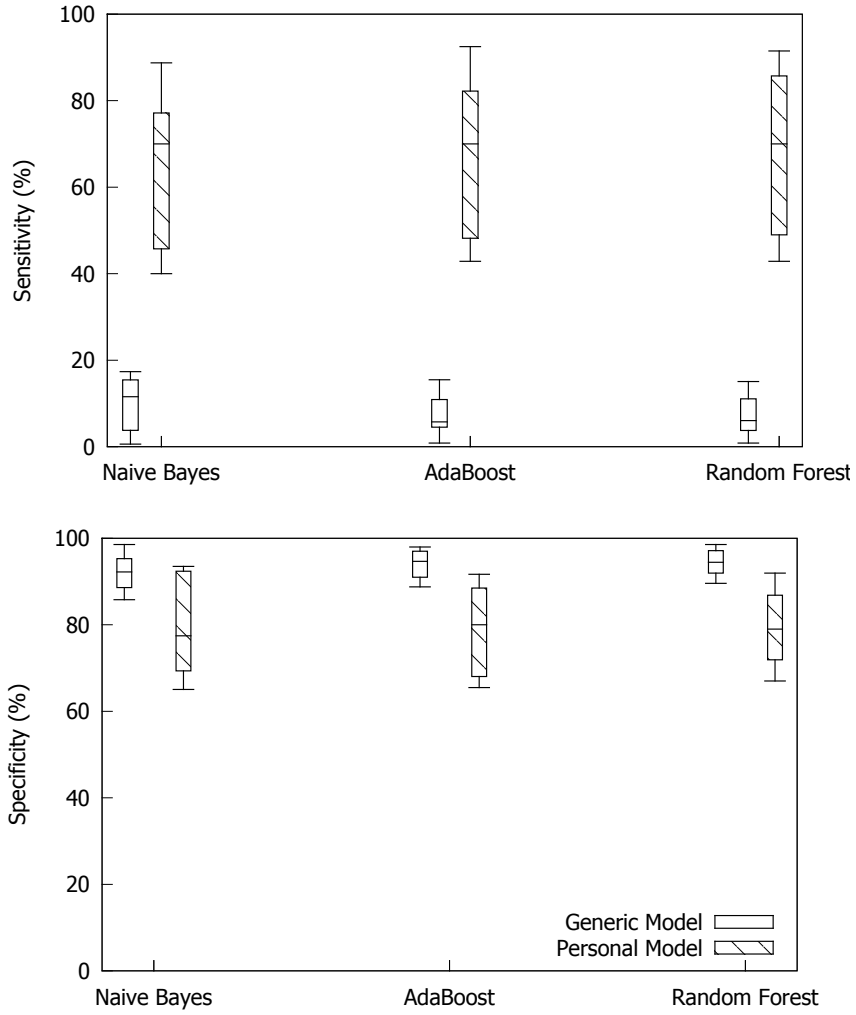


Figure 4.8: Prediction results of the generic and individual-based models.

is worth noting that for some users the sensitivity value is around 89%. This demonstrates that some people have fairly regular behavioural patterns. The predictions for the behaviour of such users are very accurate due to lower uncertainty in their behavioural patterns.

4.7 Generic vs Personal Behavioural Model

In this section we compare the performance of the prediction models trained on a user's personal data with a generic prediction model trained across multiple users' data. We build both individual-based models and a generic model for predicting notification accep-

tance by using three different algorithms: NaiveBayes, AdaBoost, and Random Forest. We trained the data-driven models in the same fashion as in the earlier section. We evaluate the individual-based models by using the k-fold cross validation approach and the generic model with the entire set of notifications of each user.

As shown in Figure 4.8, the individual-based model outperforms the generic model especially in terms of sensitivity. At the same time, the generic model achieves extremely high specificity (i.e., around 95%), because it gets trained with most of the rejected notifications in different contexts. Thus, the generic model predicts most of the notifications as declined. An interruptibility management system using such a prediction model will mostly not allow any notification to be triggered. This will, essentially, be the same as switching off the phone. To summarise, these results demonstrate that a prediction model trained on a user's personal data is more accurate for predicting interruptibility of that user than a generic prediction model trained on multiple users' data.

4.8 Online Learning Approach

The predictors discussed in the previous section are based on batch learning, i.e., they are trained on static data. Such an approach has two key drawbacks when used on mobile phones. First, the data becomes available gradually as the new notifications arrive and, therefore, a personalised model cannot be trained until a sufficient amount of data is available. Second, the prediction accuracy can be dramatically reduced when a user changes his/her behavioural pattern. Moreover, data are not usually available for new applications installed by users. For example, when a user starts using a new email client or a social networking application, the model might fail to make any accurate predictions for the notifications triggered by these applications.

To overcome these drawbacks we can use an online learning approach in which the models are periodically trained with the available data that is gradually being collected.

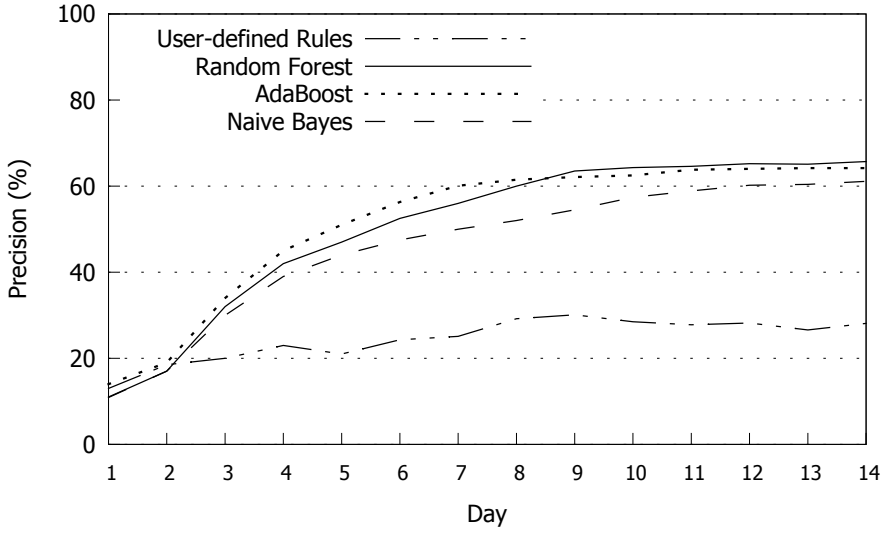


Figure 4.9: Prediction results of different predictors using online learning approach.

Such an approach enables relatively fast learning for new applications (i.e., predictions can be made in the initial days), adaptation to changing or new behaviour and improvement of the performance over time (i.e., the average prediction accuracy can be enhanced gradually as more and more data becomes available in case the behavioural patterns do not show strong variations over limited amount of time).

We use our labeled dataset to compare the prediction accuracy of different algorithms by using an online learning approach. We iteratively build all the prediction models with the notification data collected by the end of each day and evaluate these models by using the following day's notifications. For example, on day N a model was built by using notifications from day 1 to N and it was evaluated to predict the acceptance of notifications that arrived on day $N + 1$. Similarly, we train the models for user-derived rules by using questionnaire responses collected by the end of each day, and evaluate these models by using the notifications of the following day.

We test the data-driven models with three prediction algorithms: Naive Bayes, AdaBoost, and Random Forest. We use user-derived rules as the baseline to evaluate the prediction results of other predictors. As shown in Figure 4.9, AdaBoost and Random Forest are the best performers. In the initial days, AdaBoost shows better performance as compared to Random Forest. However, the prediction accuracy of AdaBoost stabilises

after seven days. On the other hand, the Naive Bayes algorithm initially shows a prediction accuracy performance comparable to AdaBoost and Random Forest, but its rate of increase in accuracy slowly declines. Overall, all the predictors outperform the user-derived rule mechanism after 2 days. It is worth noting that the top two predictors show the best performance after 9 days of training.

4.9 Discussion

According to our measurements, the users in our study receive around 100 notifications each day on average. These notifications arrive at different times of the day without any knowledge about recipients' willingness to be interrupted. Therefore, such notifications have a potential to convert the smartphone from an information-awareness device to an information-overloading one. At the same time, our results show that the response to a notification can be predicted. We believe that a user's predicted response can be used to ensure the delivery of notifications at opportune moments that might reduce perceived disruption.

To predict the response to a notification we need to build a user's behavioural model and train it with the notifications of various categories that arrive in different contexts. It is possible to capture the user's context via the sensors embedded in the phone. On the other hand, in order to categorise notifications we need to analyse their content. Some notifications can be classified by using the knowledge about the category of applications that triggered them. For example, an application in the news category is very likely to trigger notifications containing news related information.

Our results show that around 70% of the notifications that a user receives belong to email and chat categories. Thus, a further subcategorisation of these two categories is needed. As proposed in this study, we can use the social circles of users to sub-categorise email and chat notifications. For example, a chat notification can be sub-categorised as "social chat" if the sender of the notification belongs to the friend circle defined by the

user. Communication applications (e.g., email and chat) allow the users to map their social ties in the social circles defined by them. One possibility is that these circles can be used by an application to categorise email and chat notifications. However, users do not create social circles on every communication application and also the reverse is true, i.e., not all communication applications facilitate users to create social circles. This raises an interesting question about how to maintain the knowledge about users' social circle that can be used to sub-categorise the chat and email notifications.

In a recent study [LFN15], authors argue that the operating system of a mobile device should be responsible for managing and delivering notifications at opportune moments. Inspired by the authors' argument, we suggest a solution to the problem, posed above, to maintain the knowledge about a user's social circle. Since the operating system has access to all the information of every application, there should be a system service (such as Google Play Service for Android OS) that can categorise all notifications with respect to the application that triggered it and an aggregated social circle of the user generated by merging social circles from different applications (especially from the social networking and calendar applications). Such a service can monitor the user's response for all notification categories in different contexts. Thus, there will be more data available to build a richer model of the user's behaviour as compared to the models created by the applications themselves. Also, we believe that this service can provide an efficient solution because we only need to train a single but richer behavioural model of a user.

4.10 Limitations

In this work we consider two cases of user's response to a notification – accept (i.e., when a user clicks on the notification to read the information contained in it) and decline (i.e., when a user ignores or does not answer the notification up to a certain time delay). However, it is possible that some notifications were misclassified as declined because they

might be actually attended to by the users on another device or they might be read and dismissed by the users because they do not require further actions. Thus, the prediction model is trained by means of incorrect data.

Moreover, the study was conducted over a short duration and with a small number of participants. This could have introduced a selection bias that makes it difficult to claim the ecological validity of this study.

We have also identified a series of aspects that deserve further investigation, probably also by means of experience sampling techniques. One issue is related to multiple notifications from the same applications (usually referred to as "stacked" notifications). How does a user react when the notifications are stacked by an application? Does a user click/ignore it because all notifications are irrelevant or because a notification (perhaps the latest one) from the stack is important/not important? Another important aspect is related to the influence of the user's co-location with other people when a notification is received. For example, a certain user sitting in a coffee shop alone might be willing to accept a notification. However, when the same user is sitting with colleagues and discussing a project, they might not accept any notification.

4.11 Summary

In this study we have presented an analysis of the impact of content on the acceptance of mobile notifications. The results of the analysis have been used to develop a novel machine learning approach that predicts the acceptance of a notification by using its content and the context in which it is delivered. Such an approach can be used by an interruptibility management system to ensure that the right information is delivered at the right time.

We have collected notifications "in-the-wild" and classified them according to the information contained in them. Our results have shown that a user's activity can impact the time delay in the response to a notification. We have evaluated the average percentage of the notifications that are accepted in a day for each notification category. Our results

have shown that the chat notifications, where the sender is a family member or a relative of the user, have the highest acceptance rate. Overall, the acceptance value of notifications vary for each category. By using the collected real-world notifications we have designed, developed and tested a series of predictors based on state-of-the-art machine learning. We have shown that the acceptance of a notification within 10 minutes from its arrival time can be predicted with an average sensitivity of 70% and a specificity of 80%. For some users the sensitivity can go up to 89%. Our approach outperforms the user-derived rules for delivering notifications on their mobile phones. Finally, we have discussed the implementation of an online predictor in order to understand the required training period for successful prediction.

CHAPTER 5

MINING USERS' PREFERENCES FOR RECEIVING NOTIFICATIONS IN DIFFERENT SITUATIONS

5.1 Overview

Previous studies have shown that users are willing to tolerate some interruptions from notifications, in order to not miss any important information [IH10]. However, their willingness is, in a sense, exploited by mobile applications as these trigger a plethora of notifications continuously [MMHP15]. Given the potentially large number of notifications, users do not accept all of them as their receptivity relies on the content type and the source of the messages [MMHP15, MPV+16]. Users mostly dismiss (i.e., swipe away without clicking) notifications that are not useful or relevant to their interests [FYB+10, SSHD+14]. Some examples of such notifications are promotional emails, game invites on social networks and predictive suggestions by applications. At the same time, past studies have shown that users get annoyed by receiving irrelevant or unwanted notifications that could result in uninstalling the corresponding application [FEW12, SSHD+14].

The above findings provide evidence that, 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 uninter-

esting or irrelevant for the user. However, until now, most of the previous studies propose interruptibility management mechanisms that leverage the concept of anticipatory computing [PM15] to predict opportune moments by using context information [HI05, FGB11, PM14b, ORN+15]. In the previous chapter we proposed an improved design of a similar kind of interruptibility management mechanism that exploits both users' context and notification content in order to predict opportune moments.

In this Chapter, we will present the design of an intelligent interruptibility management mechanism that *learns the types of information users prefer to receive via notifications in different situations by analysing the combinations of notification titles and context modalities including activity, time and location.* Another important aspect that we are going to consider is usability: in order to tackle this problem we present the implementation of a mechanism for mining association rules [AIS93] and *making the discovered rules available to users so that they can check their appropriateness.* Moreover, unlike previous interruptibility studies, we do not restrict ourselves to evaluating the interruptibility management mechanism on offline collected datasets or through synthetically generated notifications. Instead, in order to evaluate the proposed mechanism, we conducted a real-world deployment to manage mobile notification *in-the-wild.*

5.2 Motivation of Key Design Choices

In this section we discuss the key motivations and design choices of our solution, namely (i) the choice of learning users' preferences with respect to different types of information delivered through notifications rather than modelling their interruptibility; (ii) the choice of implementing an algorithm based on learning association rules instead of alternative machine learning algorithms, such as SVM and Random Forest.

5.2.1 Why Mine User's Preferences for Different Types of Information?

The key objective of an interruptibility management system is to deliver the right information at the right time. Most of the previous interruptibility studies have proposed different approaches to model interruptibility for predicting opportune moments [HI05, FGB11, PM14b, MMHP15]. However, these studies do not provide any suggestion about what to do with notifications that arrive at inopportune moments. Should we defer them? Or should we completely dismiss them?

As discussed earlier, in a previous study [Cla96] 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). Based on Clark's suggestions *we hypothesise that an interruptibility management mechanism should take an orthogonal but equally important approach by learning the different types of interruptions that users explicitly refuse by dismissing notifications.* In this way such a mechanism can identify the notifications that are not useful for the users and stop the operating system from triggering alerts for these types of notification. Moreover, our hypothesis is in line with the findings of Fischer et al. [FYB+10, SSHD+14], i.e., users' receptivity relies on the usefulness of and their interest in the delivered information. We would like to stress again that this is an orthogonal mechanism that can be used in conjunction with others, for example for deferring the notifications until the right time/context.

5.2.2 Why Mine Association Rules Over other Types of Machine Learning Models?

One of the major issues in designing interruptibility management systems is related to their testing and evaluation. In fact, if the notification mechanisms are not correct, they might lead to deferring important notifications. Therefore, in order to build *usable* systems, we have to involve users to adjust the interruptibility management mechanisms. The goal is *to reduce interruptions without compromising useful and important information.*

The interruptibility management mechanisms proposed in the previous studies rely on machine learning models for making predictions that might be difficult to understand and translate directly into human-readable rules [HI05, PM14b, MMHP15]. These models are effective for learning quickly with a high accuracy but it is nearly impossible for users to understand these models and provide their feedback. Therefore, *we rely on mining association rules that can be easily understood by users as compared to other prediction models*. This allows us to get feedback from users about the rules that should not be used for stopping notifications on their phones. It is worth noting that a simple machine learning technique such as the decision tree [SH77], which uses a branching method to illustrate every possible outcome of a decision, is likely to find a few more rules than the association rule mining algorithm. However, the decision tree-based rules mostly have low reliability because they refer to very small sets of data instances [Ord06]. Whereas, non-reliable rules can be controlled in association rules by the means of its parameters such as support and confidence.

5.3 Mining User Preferences

In this section we discuss how we extract notification rules by mining association rules based on different combinations of notification titles and context modalities including activity, time and location.

5.3.1 Removing Reminder Notifications

The first step consists in identifying a particular class of notifications that are always dismissed but that should be shown to users in any case. As discussed earlier, notifications are dismissed if they are not found to be useful or relevant to the user's interest [FYB+10, SSHD+14]. 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* in the rest of the chapter. Alarm, calendar event and battery status notifications are some common examples of reminder notifications.

More formally, in order to define reminder notifications, we introduce a simple definition of *click rate (CR)*:

$$\text{CR} = \frac{\text{Number of accepted notifications}}{\text{Total number of notifications}} \cdot 100 \tag{5.1}$$

If an application's click rate (CR) is zero then all notifications from that application are treated as reminder notifications.

5.3.2 Notification Classification

In order to model users' preferences with respect to receiving different types of information, we categorise each notification based on the information contained in it. A recent study proposes an approach for modelling interruptibility by using information type, social circle and context information [MMHP15]. The authors of [MMHP15] assume that all notifications triggered by an application are of the same type and categorise them by using the type of application. In other words, they classify notifications at an abstract level, e.g., chat, email, systems and so on.

Instead, we do not limit ourselves to categorising a notification based on the type of application that triggered it because an application can generate notifications that contain different types of information. A user might be interested to receive some but not all types of notifications triggered by a specific application. For example, a Facebook notification about a new post on a user's timeline might not be considered to be as disruptive as a game invite notification. In other words, users would not want an interruptibility management system to completely stop Facebook notifications, but rather only those that are annoying, such as game invites.

Therefore, in order to classify notifications, we perform clustering by considering their titles by means of DBSCAN [EKSX96], a density based algorithm[1]. A notification title is

[1]We use this algorithm because it is based on an unsupervised learning approach (i.e., it does not

a short sentence that gives a glimpse of the information contained within it. The following are some examples of notification titles: *"Sign in to a Wi-Fi network"*, *"Time to Work"*, *"Today is Alice's birthday"*. It is worth noting that in some cases the notification title contains the sender name along with other text (such as *"Alice commented on your post"*) or merely the sender name (such as *"Alice"*). Generally, the sender name is attached to the titles of notifications triggered by chat and online social networking applications.

The clustering of notifications is carried out through the following steps: (i) cleaning notification titles; (ii) constructing a classifier; and (iii) clustering notifications. We discuss the details of each step in detail below.

Cleaning Notification Titles

Notification titles are short sentences phrased in a way that they are easily understood by users. The most important step for analysing these titles is to first clean them in order to remove the non-informative data.

To analyse the notification titles we follow the standard process of cleaning text in the following way:

(i) *Conversion of the the text to lower-case*: this ensures that the lower-case and the upper-case versions of the same word are considered the same.

(ii) *Removal of punctuation and numbers*: these elements of the text do not contain useful information for the classification task, and, at the same time, they might add unwanted noise.

(iii) *Removal of stop words*: stop words are common words (such as *'a'*, *'the'*, *'is'* and *'are'*) that generally have quite high frequency in the text. We remove them to ensure that they do not affect *content-bearing keywords* in the clustering algorithm [BK10].

require users to provide labelled data for training the model). Moreover, we are aware that notification clustering could be improved by using the entire notification content (i.e., including also the header of notifications). However, due to privacy concerns the notification headers were not recorded in the dataset we used in our evaluation [MPV+16].

(iv) *Removal of the sender and application names*: we remove sender and application names because they can lead the classifier to cluster notifications based on the sender or application names. Therefore, for each notification we remove (if present) the name of the application by which it is triggered. Moreover, to remove sender names we find and remove all the words that are not present in the english dictionary that comes with the `qdapDictionaries` package [qda16]. It is worth noting that in the case of chat and email notifications that contain only names, we do not remove names and thus allow the algorithm to classify such notifications based only on the sender names.

(v) *Stemming of words*: this is a standardisation method to avoid having multiple versions of words referring to the same concept by reducing a word down to its root. For instance, the words *'comment'*, *'commented'*, *'comments'* and *'commenting'* are all stemmed to *'comment'*.

Constructing a Classifier

In order to train the classifier, we use a bag of words approach [MN+98, Joa02] and create a *Document-Term Matrix (DTM)* – a matrix that describes the frequency of terms (i.e., stemmed words) that occur in a collection of documents (i.e., notification titles in our case). In a DTM, rows correspond to documents in the collection and each term is associated with a column.

To prevent the problem of overfitting the classifier [YP97], we compute the term frequency (TF) and remove the terms that have a TF lower than $TF_{threshold}$. The TF and $TF_{threshold}$ are defined as following:

$$\text{TF} = \frac{\text{Number of notifications in which the word occurs}}{\text{Total number of notifications}} \tag{5.2}$$

$$\text{TF}_{\text{threshold}} = \frac{\text{Number of participation days}}{\text{N} \cdot \text{Total number of notifications}} \tag{5.3}$$

According to the above equation the value of $\mathrm{TF}_{\mathrm{threshold}}$ ensures that notifications containing the term occurs at a rate of at least one every N days.

Finally, a DBSCAN-based classifier is constructed by using the above DTM. To cluster the data the classifier requires two parameters as inputs: (i) MP, defined as the minimum number of points required to form a dense region and (ii) ϵ, defined as the maximum difference (i.e., number of non-matching words) between the notification titles of a cluster.

Clustering Notifications

Since the notification titles are relatively short sentences, it is possible that notifications from different applications might contain similar words that could lead the classifier to cluster them together. In order to prevent this problem, for each application we create a separate classifier by using just the notifications generated by that application. For example, if there are 15 notifications from 3 applications, we create 3 clustering models and each model is trained using notifications of the separate applications.

5.3.3 Constructing Association Rules

In order to discover rules about the user's preferences for receiving notifications, we use the AIS algorithm [AIS93] – a method for mining data to discover statistical relationships between variables. The algorithm scans the data to find the frequent item sets and computes their support value (discussed later in this section). Finally, it filters the list of item sets whose support value is greater than the given support threshold.

An association rule is represented as $X \rightarrow Y$, where X is defined as the antecedent and Y as the consequent. The algorithm generates rules with the consequent containing only one item. This means that rules can be in the form of $X_1 \cup X_2 \rightarrow Y$ but not in the form of $X \rightarrow Y_1 \cup Y_2$.

To better understand the concept of association rules let us consider an example where the user: (i) always dismisses Twitter notifications about suggestions of accounts to fol-

low; (ii) accepts Facebook birthday reminder notifications only in the morning while they are at home; (iii) does not accept WhatsApp notifications from Alice while at work. Assuming that notifications about the Twitter suggestion, Facebook birthday reminders and WhatsApp messages 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:

$\{N_1\} \rightarrow \{Dismiss\}$

$\{N_2, Home, Morning\} \rightarrow \{Accept\}$

$\{N_2, Home, Afternoon\} \rightarrow \{Dismiss\}$

$\{N_2, Home, Evening\} \rightarrow \{Dismiss\}$

$\{N_2, Home, Night\} \rightarrow \{Dismiss\}$

$\{N_2, Work\} \rightarrow \{Dismiss\}$

$\{N_2, Other\} \rightarrow \{Dismiss\}$

$\{N_3, Home\} \rightarrow \{Accept\}$

$\{N_3, Other\} \rightarrow \{Accept\}$

$\{N_3, Work\} \rightarrow \{Dismiss\}$

For an association rule $X \rightarrow Y$, we define the two parameters:

- *Support*: the ratio between the number of times X and Y co-occur and the number of data-instances present in the given data. It can be represented as the joint probability of X and Y, i.e., $P(X, Y)$.
- *Confidence*: the ratio between the number of times Y co-occurs with X and the number of times X occurs in the given data. It can be represented as the conditional probability of X and Y, i.e., $P(Y \mid X)$.

An association rule is created only when it has at least the minimum support (S_{min}) and confidence (C_{min}). It is worth noting that decreasing the threshold values of either support or confidence could result in discovering more rules [AIS93].

5.4 Evaluation of the Rule Learning Mechanism

In this section we discuss the implementation and evaluation of the mechanism for mining individual-based association rules about users' preferences.

5.4.1 Dataset

In order to evaluate the proposed solution, we use the data collected by means of smartphones during the My Phone and Me study presented in Chapter 3. From the original dataset, we took a subset of data considering only the users who participated for at least 14 days. Moreover, the dataset contains an attribute to identify whether a notification is clicked, dismissed or handled on another device. We did not consider notifications that were handled on other devices. Consequently, the final dataset used in our analysis contains 11185 notifications from 18 users.

5.4.2 Evaluation Settings

We evaluate the discovered rules for predicting users' response to notifications by using a k-fold cross validation approach with the value of k equal to 10. The prediction results are computed separately for each user and we aggregate them by computing the mean and the standard-error with a 95% confidence interval. We divide the set of notifications belonging to each user into a training set and a test set, where the training set contains 90% of the data and the rest is considered as the test set.

5.4.3 Defining Configurations

As discussed earlier, we use the $TF_{threshold}$ to prevent the problem of overfitting the DBSCAN-based classifier with sparse (i.e., infrequent) terms. In order to find an optimal value of $TF_{threshold}$ we create a DTM for all notification titles in the dataset and compute the number of terms filtered by setting $TF_{threshold}$ with different values of $N \in [1, 7]$. Here,

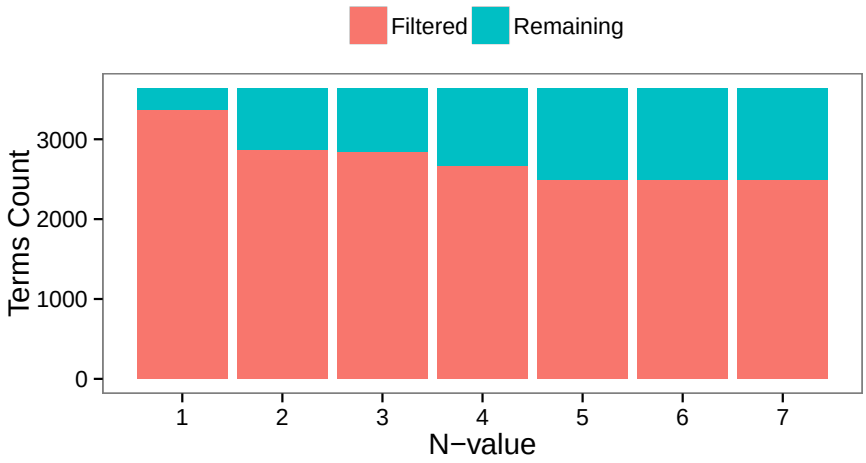


Figure 5.1: Filtered and remaining terms for notification clustering with different values of N in $TF_{threshold}$.

we do not consider values of N greater than 7 because we believe it would not be useful to include notification terms that are not received at least once in a week.

As shown in Figure 5.1 there are 3642 unique terms in the dataset. By setting $N = 1$, there are chances of under-fitting the model because 3371 terms are removed and only 271 terms remain in the DTM. Since there is limited difference in the number of filtered terms with $N \in [2, 7]$, we use $N = 2$ for our analysis, which also ensures that each term is seen by the user at least once in two days.

In order to satisfy the above constraint, for each user we set the value of MP to $D/2$ (D indicates the number of participation days). This ensures that there are at least $D/2$ notifications in each cluster. Similarly, we set S_{min} (i.e., minimum support value) equal to $D_{participate}/(2 \cdot N_{count})$, which ensures that the notification interaction pattern covered in each rule has occurred at least once in two days. Here, $D_{participate}$ indicates the number of participation days of the user and N_{count} refers to the total number of notifications collected for that user. Moreover, we set ϵ to 1 so that each notification title will have at least $N-1$ (N refers to the number of words in a title) words similar to other notification titles in its cluster.

5.4.4 Feature Selection

In order to discover association rules about the user's preferences we rely on the following features:

(i) *notification response*: the user's response (i.e., click or dismiss) to a notification;

(ii) *notification type*: the identifier of the cluster to which the notification belongs;

(iii) *arrival time*: the arrival time of the notification considering four time slots – morning (6-12), afternoon (12-16), evening (16-20) and night (20-24 and 0-6);

(iv) *activity*: the user's physical activity (includes still, walking, running, biking and in vehicle) when the notification arrived;

(v) *location*: the user's location when the notification arrived.

Note that we do not include the alert modality of notifications for mining rules because we are interested in finding the information that users prefer to receive via notifications in different situations irrespective of their alert modalities.

By using different combinations of these features we construct the association rules according to the following five approaches:

1. **AR1**: by using notification response with notification type;
2. **AR2**: by using notification response with notification type and activity;
3. **AR3**: by using notification response with notification type and arrival time;
4. **AR4**: by using notification response with notification type and location;
5. **AR5**: by using notification response with notification type, activity, arrival time and location.

For all approaches we restrict the consequent 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 in predicting the acceptance of a notification, therefore other items in the consequent would be of no use and just add extra computational load.

5.4.5 Prediction Results

In this section we present the accuracy of the association rules discovered with different values of C_{min} for all five approach. In order to assess the discovered association rules, we compare the predicted response with the actual response (i.e., the ground truth) and compute the accuracy in terms of:

- *Recall*: ratio between the number of notifications that are correctly predicted as dismissed and the total number of notifications that are actually dismissed.
- *Precision*: ratio between the number of notifications that are correctly predicted as dismissed and the total number of notifications that are predicted (both correctly and incorrectly) as dismissed.

In Figure 5.2 we present the prediction results for the association rules constructed by using all five approaches. 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.

The association rules constructed with approaches *AR1* and *AR2* do not show a significant difference. The recall of *AR2* is slightly higher when the confidence is below 70%, but the precision drops as well, which means some extra but unreliable rules are discovered. On the other hand, the association rules constructed with *AR3* achieves better recall than *AR1* and *AR2* but its precision consistently remains lower (i.e., under 85%) compared to other approaches. This implies that there is no significant contribution of the activity and arrival time features in terms of predicting user's preferences for receiving notifications.

The approach *AR4* (i.e., the association rules that are constructed by using the notification response, type and location) performs better that other approaches in terms of both recall and precision. Its recall goes up to around 43% without reducing the precision below 79%. Even by combining all features together in the approach *AR5*, the results do not improve. As compared to approach *AR4*, there is a negligible increment in the

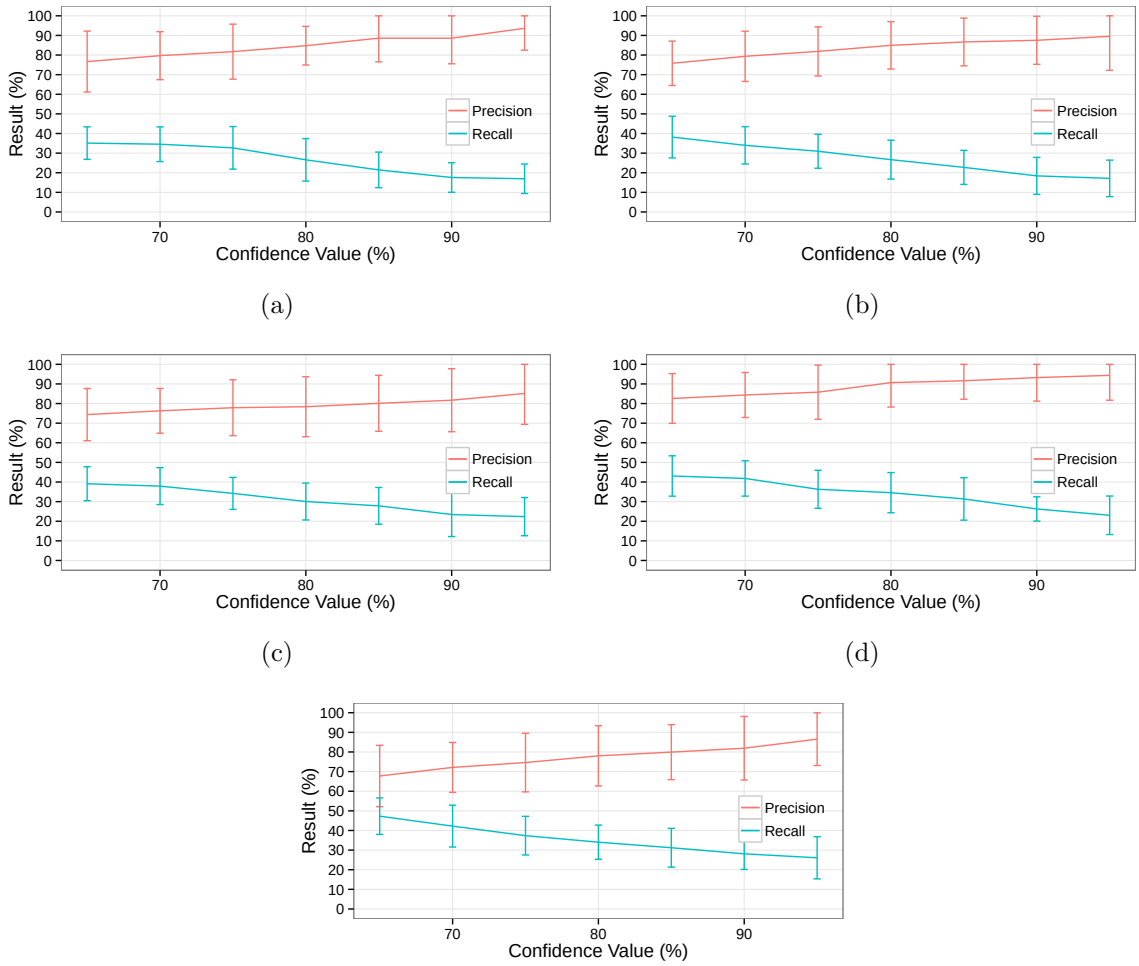


Figure 5.2: Prediction results for association rules discovered by using notification response along with: (a) AR1: notification type; (b) AR2: notification type and activity; (c) AR3: notification type and arrival time; (d) AR4: notification type and location; (e) AR5: notification type, activity, arrival time and location.

recall of the approach *AR5* with a big decrease in its precision. Consequently, our results provide evidence that the type of information contained in a notification and the location of the user can be effectively used to learn user's preference for receiving notifications. However, this is not the case for users' activity and notification arrival time, at least for our dataset.

It is worth noting that the high standard-error in the results demonstrates that our mechanism does not work consistently for all users. This is due to the fact that users have different preferences for receiving notifications. Moreover, the upper-limit of the standard-

error for precision close to 100% demonstrate that for some users our mechanism is able to very accurately predict notifications that are not interesting or useful for them.

Another interesting observation is that the recall of approach *AR3* is always higher than that of *AR1* and *AR2*, but its precision is instead always lower. We believe that *AR3* attains higher recall because users follow precise circadian rhythms and during different time periods of a day they are present at certain locations (e.g., at work in the afternoon and at home during night) [EP09]. Therefore, in *AR3* rules are essentially associated to their behaviour at certain locations rather than the time of notifications' arrival. Moreover, the precision might be affected as users' mobility pattern might vary on some days.

5.4.6 Optimising the System for High Precision

The key requirement for an interruptibility management mechanism that would be not only useful but also acceptable to users 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) that could be achieved by ensuring that the precision remains close to 100%. However, we could consider a precision of around 90% as acceptable because it might be possible that a couple of notifications have been clicked by mistake or to kill time when the user was bored.

At the same time, the interruptibility management mechanism should also achieve a significant recall in order to prove 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. Though we have already filtered out reminder notifications, there could still be notifications from other applications which are useful, but do not require any action, such as a chat message like *"See you at 7pm"* and a confirmation email like *"Ok, bye!"*.

We observe that, by using the confidence of 70%, the approach *AR4* is able to achieve around 42% recall with a precision of 84%. However, the reduction of 16% in terms of precision might lead the mechanism to predict some false-negatives. So, we should compromise by having around 35% recall with a precision of more than 90% at a confidence of 80%. Here, even the 35% recall implies that we are able to filter a big portion of non useful notifications and thus reduce disruption.

5.5 Online Learning

In the previous section we evaluated our mechanism by using the batch learning method in which the association rules are mined by using static data. However, in a real world scenario such training data is not initially available and, therefore, association rules cannot be discovered until a sufficient amount of data has been collected.

In order to address this issue, we can use an online learning method (discussed in Chapter 4) in which the association rules are periodically mined from the data. Even if this solution does not completely remove the problem of initial bootstrapping, the prediction accuracy improves gradually as more and more training data becomes available. Moreover, by using an online learning method a system can adapt itself according to the potential changes in user behaviour over time. However, the model would not be able to adapt and might also perform worse when there are frequent changes in the user behaviour.

In order to evaluate this method, we iteratively construct association rules with all the notifications collected by the end of each day and evaluate these rules by using notifications of the following day. For example, on day N a model is built by using notifications from day 1 to $N-1$. Moreover, similarly to the evaluation used with the batch learning method we configure S_{min} as $(N-1)/(2 \cdot N_{\text{count}})$, where N_{count} indicates the total number of notifications collected until day $N-1$. This ensues that the notification interaction pattern covered in each rule has occurred at least once in two days or at least $(N-1)/2$ times in $N-1$ days.

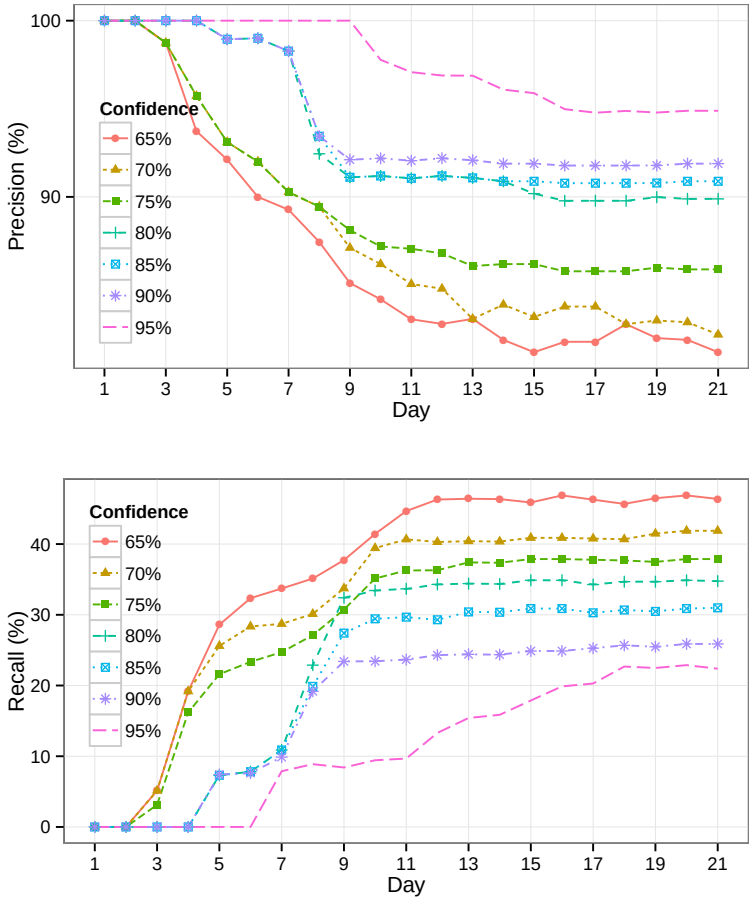


Figure 5.3: Prediction results for association rules (with notification title and location) using the online learning approach.

Figure 5.3 shows the prediction results for the individualised rules iteratively constructed on each day with different values of C_{min} by using the online learning method. Rules with C_{min} as 80 and below start filtering notifications from the 3^{rd} day and by the 7^{th} day they achieve recall of 25-35% and precision around 90%. Instead, by the 7^{th} day other rules could achieve recall of 10% and precision above 90%. Interestingly, rules with C_{min} as 95% never become stable. This could be due to the fact that these rules filter notifications with high reliability which can be confirmed by their precision always remaining over 90%. On the other hand, rules with C_{min} as 65-70% and 80-90% become stable by the 12^{th} and 9^{th} day respectively.

It is worth noting that given the duration of the dataset, it is not possible to evaluate the adaptation of the algorithm and confirm if the set of rules are stable over time. It might be possible that users diverge from their notification interaction behaviour and the system needs to adapt accordingly. Therefore, we envisage that the system should use a sliding window approach, i.e., the algorithm should be trained on the last L days in order to be able to adapt to changes in user's behaviour.

5.6 MyPref Library

5.6.1 MyPref at a Glance

Starting from the mechanisms and evaluation presented in the previous section, we funnelled our findings into the implementation of the MyPref library – an intelligent interruptibility management library that can predict the type of notifications that users would prefer to receive on their mobile phones in specific contexts. The library learns the user's preferences for receiving notifications by mining association rules with the notification and context data that is supplied to it and predicts the user's receptivity to the subsequent notifications. The MyPref library 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 into 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. The abstractions include clustering of notifications, mining association rules and predicting the acceptance of notifications. The library relies on Weka for Android [Wek16] and the Snowball stemming library [Sno16] for clustering notifications locally on the phone. Since the computation is performed locally,

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

the library also preserves user's privacy since no data is transmitted to a back-end server. It offers flexibility to developers by allowing them to define notification clustering configurations (i.e., $TF_{Threshold}$, ϵ and MP) as well as configuring the rule mining parameters (i.e., support, confidence and features to be used for mining rules).

The library learns the user's preferences for receiving notifications and returns the discovered rules as output. In order to make the prediction mechanisms more transparent to users of the applications built on top of the MyPref library, it makes the rules human-understandable by replacing each notification type with the most frequent words[1] of the relevant notification cluster (we refer to these words as *keywords* in the rest of this chapter).

For instance, let us consider the following examples of hypothetic *keywords* in notification clusters: 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"*. The keyword from the cluster of Google Play Store's app update notifications (such as *"2 applications updated."* and *"3 updates available."*) would be *"update"*. The keywords from the cluster of systems WiFi availability notifications (such as *"Wi-Fi networks available"* and *"Verizon Wi-Fi available"*) would be *"Wi-Fi"* and *"available"*.

The overlying application can present these rules to users in order to ask for their consent and use the rules that are accepted by them to filter notifications. Later in this section, we will show a potential approach to ask for confirmation of users for applying the predicted rules in an application.

Moreover, producing rules with keywords instead of cluster identifiers would reduce the computation time at the arrival of notifications. Indeed, this requires the clustering of notifications in order to find their cluster identifiers and then predict the response by using the rules. This might be fine for applications that are predicting the acceptance for their own notifications. However, if an application is managing notifications from third-

[1]We consider the most frequent words after removing all the stop words and stemming the remaining words.

party application then the prediction process should be very quick so that, if required, it can cancel a notification before it alerts the user (via sound, vibration or LED light).

5.6.2 Evaluation of the MyPref Library

In this section we discuss the performance of the MyPref library. We use a Nexus 6 phone with 3 GB of RAM and a quad-core 2.7GHz Krait 450 CPU, running a clean slate Android 5.0 OS for the performance evaluation of the library.

Source Code and Memory Footprint

MyPref is implemented as a light-weight Android library consisting of 15 Java classes built with 4077 lines of code. We developed a stub application on top of the library to evaluate the memory footprint that accounts to 7.559 MB (including Weka and Snowball stemming libraries). We used third-party measurement tools, namely Count Lines of Code [CLO16] and Android Dalvik Debug Monitor Server [DDM16], to evaluate the source code and memory footprint of the library.

Energy Consumption

A fundamental aspect of the design of this class of libraries is the energy usage and its impact on the phone's battery life. We analyse the energy consumption of the library by varying the amount of data in the input. We characterise the battery charge consumption for mining association rules for all approaches (*AR1*, *AR2*, *AR3*, *AR4* and *AR5*). We use Power-Tutor [ZTQ+10] for taking the battery measurements.

As shown in Figure 5.4, the battery consumption increases almost linearly as the amount of data used for mining the rules increases. However, as the number of features increases the energy consumption increases but not dramatically. This implies that most of the energy is consumed for clustering the notification types and less energy is required for mining the rules.

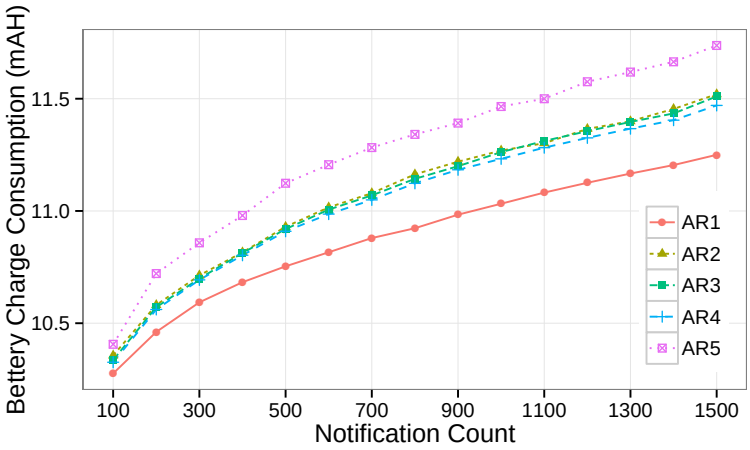


Figure 5.4: Battery charge consumed for mining rules using different approaches with different number of notifications as an input.

Moreover, the battery consumption varies slightly for approaches *AR2*, *AR3* and *AR4* if they use the same number of features. The reason for this difference could be that these approaches use features that have a different number of classes. For instance, location has three classes: *home*, *work* and *other*, whereas activity has five classes: *still*, *walking*, *running*, *biking* and *in vehicle*. Increase in the number of classes for a feature would require more iterations for discovering rules and thus consume more battery.

Time Complexity

We compute the time required for mining association rules by using all five approaches for different amounts of training data. The time complexity of the algorithm is $\mathcal{O}(N_{count})$.

As shown in Figure 5.5, as expected, the time taken increases linearly as the amount of data increases. Quite interestingly, the difference in terms of time required for the computation (for the various approaches taken into consideration) increases as the number of notifications increases. Moreover, it also increases as the number of features used gets larger. For instance, the library takes less than 61 seconds when mining association rules from 1500 notifications for any approach.

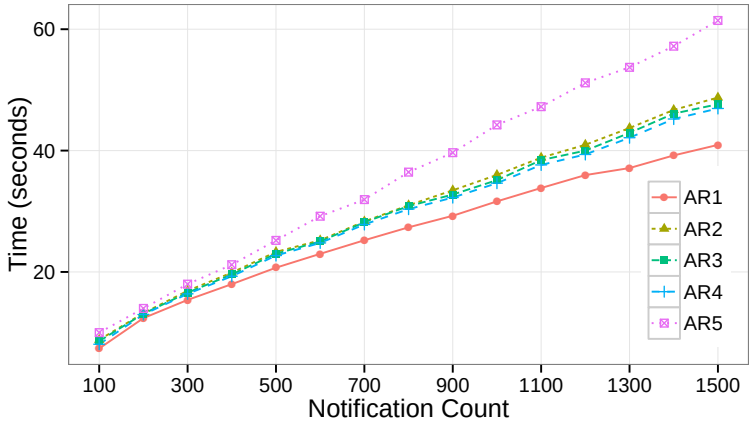


Figure 5.5: Time taken for mining rules using different approaches with a varying number of notifications in input.

5.7 *In-the-wild* Evaluation

In this section we present an *in-the-wild* evaluation of PrefMiner (see Figure 5.6), a mobile application that is able learn the user's preference for receiving different types of notifications and filtering out the notifications that are not useful, uninteresting and irrelevant to the user.

The PrefMiner application is built on top of the MyPref library discussed and evaluated in the previous section and uses the notification type and location data (i.e., the approach *AR4* as discussed above) for mining the association rules. 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 5.6.a the newly discovered rules are presented to the users in a human-readable format to get their consent. 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. Some possible examples of such rules are the following: *"Stop notifications from Facebook that contain 'candy' and 'crush' words in the title."* and *"Stop notifications from WhatsApp that contain 'Alice' words in the title and arrive at WORK."*

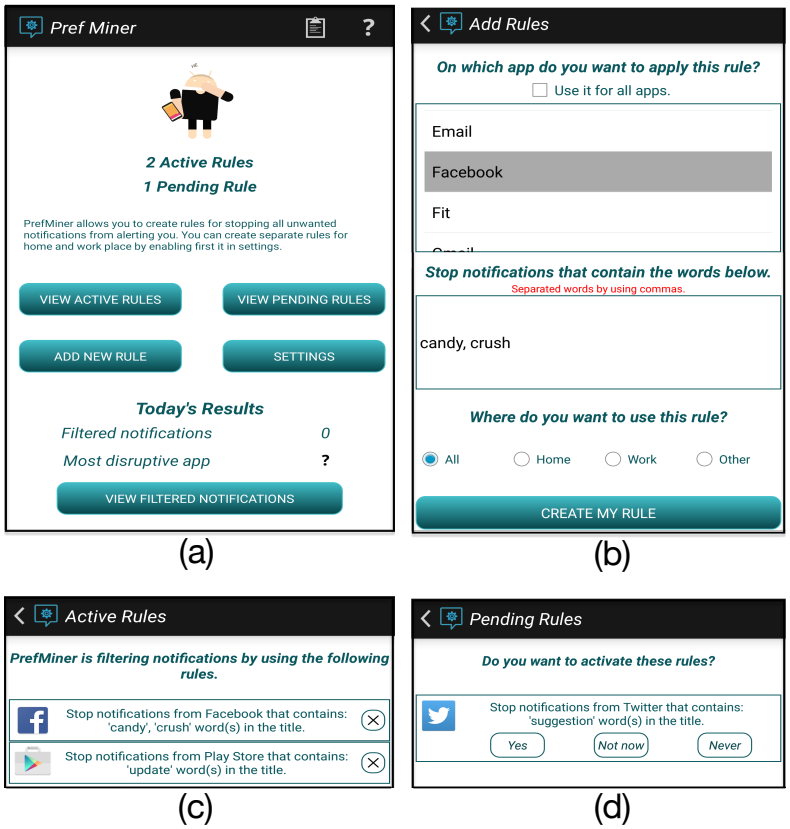


Figure 5.6: PrefMiner application: (a) main screen, (b) self-rule creation, (c) active rules, (d) pending rules.

Users can accept the rules, which they think are correctly discovered, by clicking *"Yes"*. If the user clicks on *"Never"* for a rule, the applications stores that rule as a blacklisted rule and never shows it again in the future. To ensure this, after every rule mining process, the application removes all blacklisted rules from the newly discovered rules. Moreover, if the user clicks on *"Not Now"*, the rule is re-proposed during the next iteration of the mining process. Finally, once a rule is accepted by a user, it becomes active (see Figure 5.6.c) and the application starts filtering out all the subsequent notifications according to the rules that are currently active. The user can click on the "View Filtered Notifications" button (shown in Figure 5.6.a) in order to view the list of filtered notifications along with their time of arrival and the triggering application.

5.7.1 Deployment of PrefMiner

The PrefMiner application was published in the Google Play Store and advertised through social media and other channels in our University. We ran the study for 15 days followed by an exit questionnaire (see Table 5.1). Overall, 18 people participated in the study without any monetary incentive. However, one user did not answer any questions about the suggested rules and another user dismissed all the rules. Therefore, we considered only the remaining 16 users when evaluating the application.

To make the application interesting for the users we allowed them to manually create their own rules. As shown in Figure 5.6.b, a manual rule can be created by defining the application (selected from a list of installed applications), keywords and location. In order to ensure that the manual rules do not affect our experiment we prevent the users from manually create any rule during the period of the study (i.e., the first 15 days after the installation of the application). Note that, for privacy reasons, the user has to give explicit permission after the installation as required by the Android operating system to allow PrefMiner to manage notifications. Moreover, to ensure that the user is aware of data collection, the application also shows a detailed user consent form.

5.7.2 In-the-wild Evaluation Results

During the study, PrefMiner suggested 179 rules to the participants out of which 102 rules (56.98%) were accepted. Figure 5.7 shows the count for accepted and dismissed rules for each user that are sorted according to their rule acceptance percentage. The graph shows that there are some users who accepted most of the rules and some who accepted only a few rules. Overall, 11 users (i.e., around 70% of the total users) accepted 50% (and above) of the suggested rules.

In order to find why and what rules were dismissed by users we analysed all dismissed rules. Our results show that most of the dismissed rules are about the "communication applications" including WhatsApp, Gmail and Hangouts. The other most dominant app

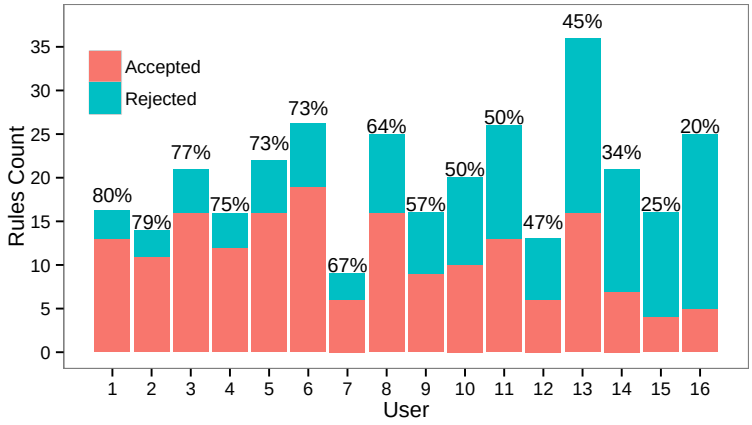


Figure 5.7: Users response to the rules suggested by the PrefMiner application.

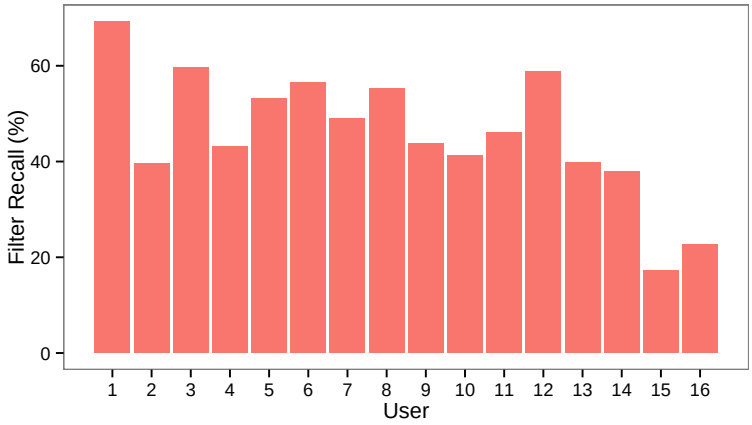


Figure 5.8: Performance of PrefMiner in filtering notifications.

category for dismissed rules is "system applications" including Bluetooth Share, Android System and Google Play Store.

During the study the application also collected the notifications that arrived on the user's phone. Our results show that each day a user on average receives around 71 notifications (with a standard deviation of 22). We compute the *filter recall* as a percentage of filtered notifications over the sum of the notifications dismissed by the users and the filtered ones (i.e., all the notifications automatically or manually filtered). The results show that the filter recall achieved by the accepted rules is 45.81%. More specifically, the average number of notifications that are successfully filtered everyday is 12 (with a standard deviation of 8). In Figure 5.8 we show the filter recall of each user. For some

Question	Average Response	Median Response
Q1. I found the app useful for learning my preferences to filter notifications.	4.25	5
Q2. The app filtered most of the notifications that I didn't want to receive.	4.33	5
Q3. The app incorrectly filtered notifications that I wanted to receive.	1.58	1

Table 5.1: Exit questionnaire with the average and median of responses from users.

users PrefMiner is able to identify around 60% of the notifications that users do not want to receive. On the other hand, the filter recall for some users is fairly low (approx. 20%). This indicates that there are differences in the individual's preferences for filtering notifications. However, it is worth noting that the system is optimised for high precision so that it does not filter out any important notifications. Moreover, for each day, on average, 7 notifications per user are classified as reminders.

Exit Questionnaire

At the end of the 15-day study, the application asked users to fill in an exit questionnaire (shown in Table 5.1) that can be used to quantify the usefulness and accuracy of PrefMiner according to the users. Users were allowed to register their response on a 5 point Likert scale (1: *Strongly disagree* - 5: *Strongly agree*). Out of the 16 participants only 12 participants registered their response to the questionnaire. Table 5.1 lists the average response to the three questions provided by the participants. The results demonstrate that users found the application useful for filtering notifications and according to them it filtered most of the unwanted notifications correctly.

5.8 Limitations

In this chapter we have presented the design, implementation and evaluation of the MyPref library that learns the user's preference and manages subsequent notifications, and also

PrefMiner, an application built on top of it. We believe that our approach for making prediction techniques transparent to users helps to build their trust since it reduces the risk of stopping any important notification by involving them in the extraction of notification preference rules. At the same time, we have shown that our solution helps users by stopping 45.81% of notifications that are not considered useful or important and, thus, minimises their disruption.

However, the proposed interruptibility management mechanism has some limitations. Firstly, it does not provide support for deferring notifications at inopportune moments. This additional feature might be useful in certain situations and it is a part of our future work. Secondly, our mechanism is not able to accurately detect reminder notifications. Instead, it finds only the applications for which all notifications are dismissed and labels them as reminder applications and reminder notifications. We believe that if we can accurately identify and remove all reminder notifications, our interruptibility management mechanism would be able to discover more accurate rules and thus the overall accuracy could be improved. A potential solution might be based on the manual annotation of certain type of notifications, such as the standard Android battery warning messages.

Another limitation is that we are not able to infer the user response to notifications that are handled from other devices. Thus, we have to discard such notifications. We believe that by being able to detect the user response for such notifications, our mechanism might be able to make correct predictions in fewer days and might also discover some interesting rules across devices.

Moreover, the study was conducted over a short duration and with a small number of participants. This could have introduced a selection bias that makes it difficult to claim the ecological validity of this study. Finally, the current implementation of the MyPref library can only support mobile phones that are configured to use the English language. This is because it relies on modules for processing text in English. However, it can be easily extended to other languages by adopting text processing libraries, such as a library for stemming, for those languages.

5.9 Summary

In this chapter we have presented a novel solution for intelligent notification management based on the automatic extraction of rules that reflect user's preferences. The goal of the proposed approach is to make notifications *intelligible* to users. We have first evaluated our proposed mechanism with a dataset of real-world notifications collected during the My Phone and Me study presented in Chapter 3. Our results show that by using the notification title and the user's location, we can predict if a message will be dismissed by a user with a very high precision.

We have also discussed the design of an open source Android library implementing the interruptibility mechanism and the implementation of the PrefMiner application built on top of it. Through an *in-the-wild* deployment, we have shown that PrefMiner could be an effective, yet transparent, solution for interruptibility management for mobile devices.

CHAPTER 6

PREDICTING THE RIGHT DEVICE ON WHICH TO DELIVER NOTIFICATIONS IN CROSS-PLATFORM ENVIRONMENTS

6.1 Overview

As discussed in the previous chapters, despite being beneficial to users for proactive information delivery, notifications sometimes arrive at inappropriate moments that can have adverse impact on the execution of the ongoing task [BKC00, CCH01, CCH00a, MBDT02] and even on the affective state of users [AB04, BK06]. 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.

We have earlier presented the design, implementation and evaluation of interruptibility management mechanisms to infer opportune moments for delivering mobile notifications and to learn the types of notifications users prefer to receive in different situations. This chapter focuses on 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 current context. For instance, let us consider a scenario where a user is interacting with their friend on Hangouts using their laptop: a notification (from

Hangouts or another application) should be shown on the laptop rather than on the phone or smartwatch.

More specifically, in this work 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 device other than their mobile phones) at the same time. We then present the design, implementation and evaluation of NotifyMeHere, a solution for intelligent notification delivery in multi-device environments. Finally, we show that users can be clustered based on notification interaction and lifestyle-based similarity measures for constructing group predictors. This type of group predictors can be used to address the bootstrap problem, i.e., to provide a baseline behaviour when an application is initially installed and there is insufficient training data.

6.2 Dataset

In order to investigate the factors that determine users' decisions for handling notifications on specific devices in different situations, we use the data collected by means of mobile phones during the MyTraces study presented in Chapter 3. From this dataset, we select 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. 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) is comprised of 57,242 notification instances, 8.19 million phone usage events and 1.9 million context samples.

6.2.1 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 device: personal phones and alternative devices (i.e., any device other than their phone). However, the

proposed model is highly generalisable to the case of multiple classes of devices that support cross-platform applications.

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 of the phone if it was delivered on some other device and the user has already interacted with it on the 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 during 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 an email notification from Gmail on their phone as well as their laptop, but they view and click the notification on their laptop then the Gmail application running on the phone automatically removes the notification from the notification bar.

Therefore, to check the user's interaction during t_a and t_r, we use the phone usage data collected by our application (see Table 3.6). As discussed earlier, this data contains the logs for clicks, changes in the foreground and lock/unlock events.

Issues with Long Response Time

As discussed in Chapter 4, some applications automatically kill the notifications that they have generated and that 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 filter out all notifications for which $t_r - t_a > 30\ minutes$.

Removing Stacked Notifications

Another issue with the data is the duplicate entries introduced by "stacked" notifications (i.e., multiple notifications from the same applications being displayed in the notification

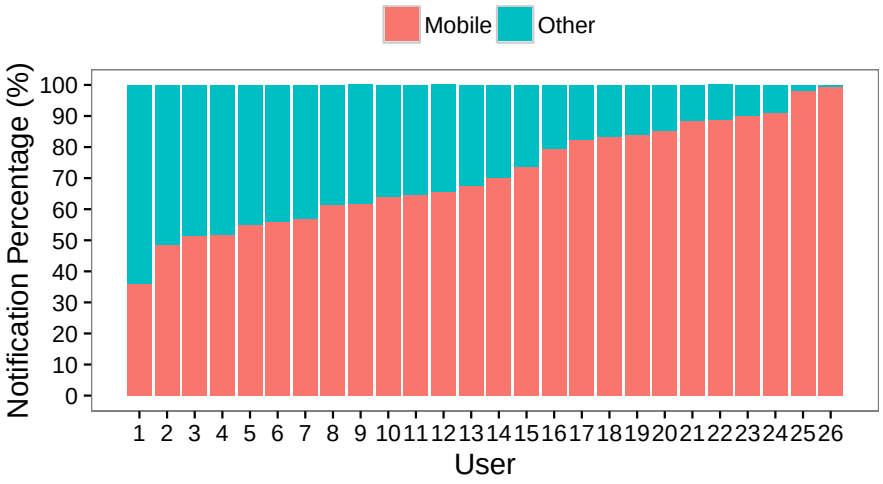


Figure 6.1: Percentage of notifications handled on mobile and other devices by the users.

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 stacked notification update and, thus, our application collected them as separate notifications.

In order 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.

Notification Distribution

In Figure 6.1 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, 71.41% of notifications 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 there without any prediction. Therefore, for fairness, we discard two users (users 25 and 26 in Figure 6.1) 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.

6.3 Impact on Notification Handling Behaviour in a Multi-device Environment

In this section we investigate the effect of different factors on users' behaviour with respect to notification handling in a multi-device environment. In order to perform this analysis, we link notifications to contextual and notification-based features that include:

- **Activity:** physical activity as inferred from phone sensors. As discussed earlier, the application relies on the Google's Activity Recognition API [GAR16] to log the information about users' physical activities classified as *walking*, *cycling*, *commuting on 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 a 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 [TM15]. 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

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 6.1: Analysis of factors influencing users' decision for handling notifications on multi-device environment.

(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 labelling might be improved by using geo-spatial data, but this is outside the scope of this dissertation. 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 application that triggered the notification. Since the categories defined by the Google Play Store are too generic, we manually categorise the applications. Here we consider only the cross-platform applications (i.e., the applications that have notifications that are handled on an alternative device) 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 behaviour 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 [Pea00] to investigate if there is a statistically significant relationship between the DV and IVs. In Table 6.1 we present the results of our analysis. It is possible to observe that there is a significant relationship between the user's behaviour 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 behaviour.

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. 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 in all places. For instance, we might carry multiple devices while at home, in the office or in a coffee shop, but not for example inside a gym. At the same time, the association with the application category demonstrates 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 or desktops mainly due to usability factors.

6.4 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:

- **Individualised model**: trained only with the data of a single user, i.e., the owner of the device.
- **Generic model**: trained with the entire dataset containing data about all users.

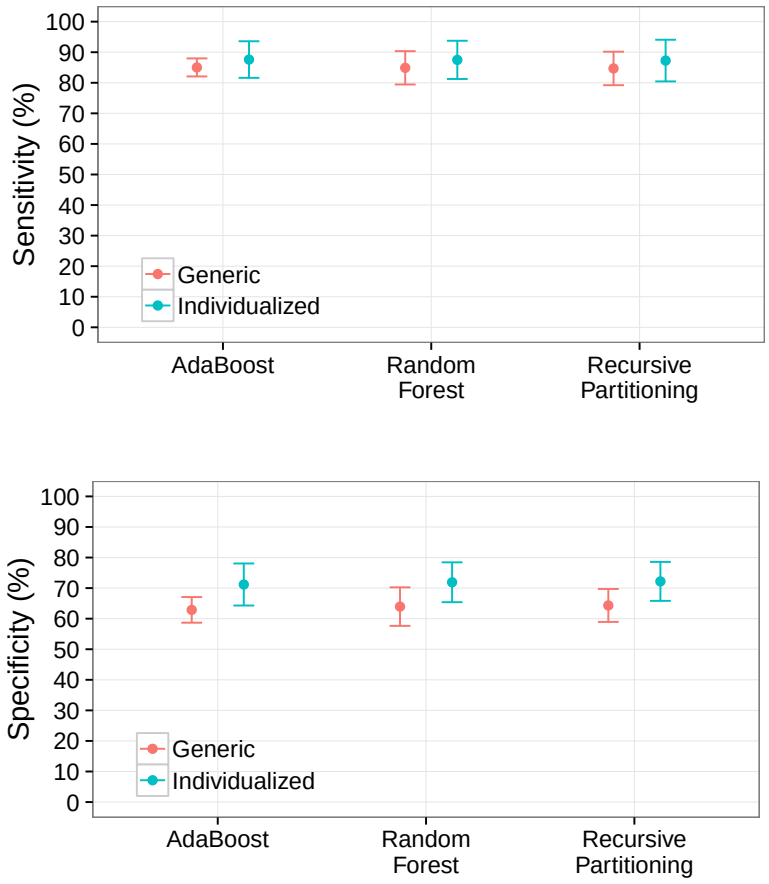


Figure 6.2: **Prediction results for individualised and generic models using three different machine learning algorithms.**

As discussed above, users' notification handling behaviour 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 [ROM01], Random Forest [Bre01] and Recursive Partitioning [Fri76]. We evaluate these models for predicting the device on which the user will handle a notification in different contexts by using k-fold cross validation approach with the value of k as 10.

Moreover, to assess 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.

In Figure 6.2 we present the prediction results for individualised 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.

6.4.1 Why are the Performance Results of the Individualised and Generic Models Similar?

In order to investigate why there are such small differences in the performance of individualised and generic models, we explore the *importance* (based on the Gini index [Bre01]) 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 not significant difference in their performances [RS04]. We opted for the Gini index as it is used internally in many classifiers some of which are used in our analysis.

As shown in Table 6.2 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

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

Table 6.2: Gini index of the features used for training prediction models.

most of the notifications are handled on the same device as their preceding notification and, thus, both types of model can achieve high accuracy by using solely the "previous click" feature.

6.4.2 Analysing Notification Handling Sequence

In order to understand users' behaviour 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 6.3, 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 [FMK+10]. For instance, if a user starts chatting with a friend on their laptop then with a high probability they will continue to chat on that device instead of switching to their phone.

6.4.3 Can We Rely on the Previous Click Feature Even When The Context Switches?

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

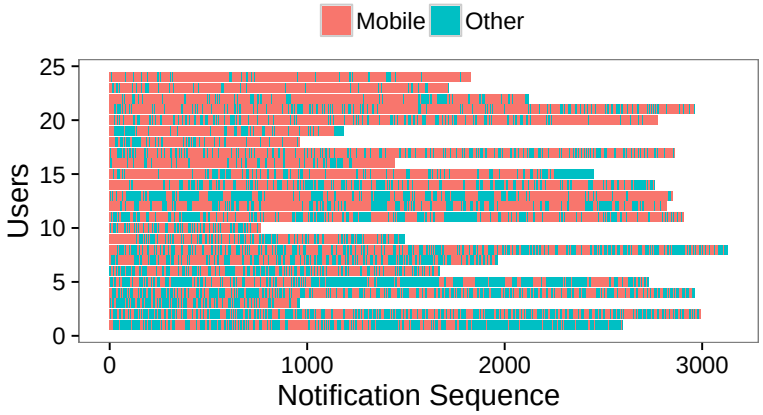


Figure 6.3: Sequence of notifications handled by users on mobile and other devices.

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 role here since users' context may change after some time. Indeed, in a new environment users 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 [WVKH16]. 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 20 seconds (which is less than than

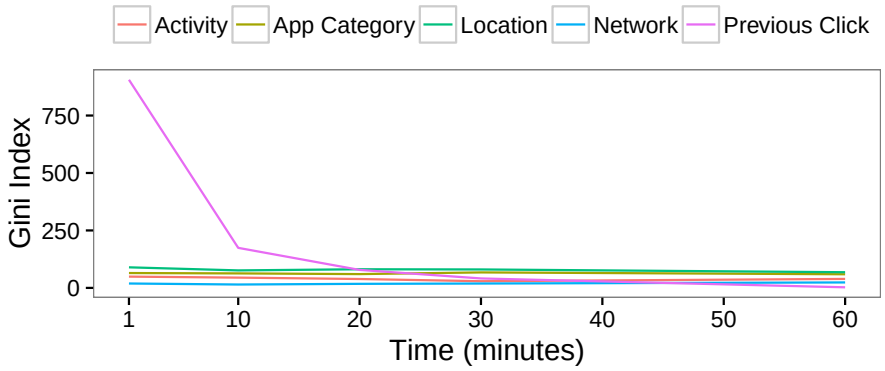


Figure 6.4: Importance of variables for predicting the right device.

$t_{threshold}$). However, n_3 is not filtered because the time gap between n_1 and n_3 is 70 seconds (here n_1 is considered as a preceding notification because n_2 is dropped out). 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 this is computed only with the training data.

Finally, we compute the variable importance by setting the value of $t_{threshold} \in$ [0, 1, 20, 10, 30, 60] minutes. Here, 0 minutes refers to no filtering of notifications. As shown in Figure 6.4, the "previous click" feature dominates over all other features only for $t_{threshold}$ 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.

6.4.4 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 in-

troduce a new feature, namely *"time elapsed - notification"* (i.e., the time 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 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 behaviour 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.

6.4.5 Exploiting Context Change Features to Improve the Prediction Models

We build both individualised 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,*

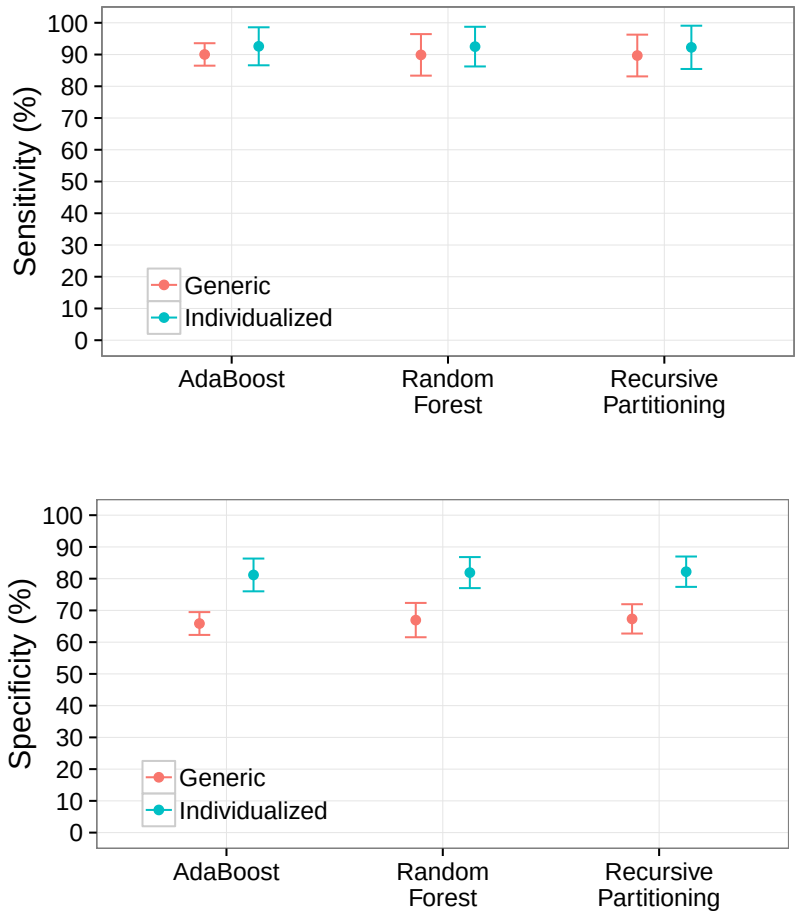


Figure 6.5: Prediction results for individualised and generic models built on three different machine learning algorithms by using new features.

previous click, time elapse activity, time elapse location, time elapse network and *time elapse notification.*

In Figure 6.5 we present the prediction results for both individualised and generic models. The results suggests that for all machine learning algorithms the individual-based models outperform the generic models in terms of specificity, but there is little 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.

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

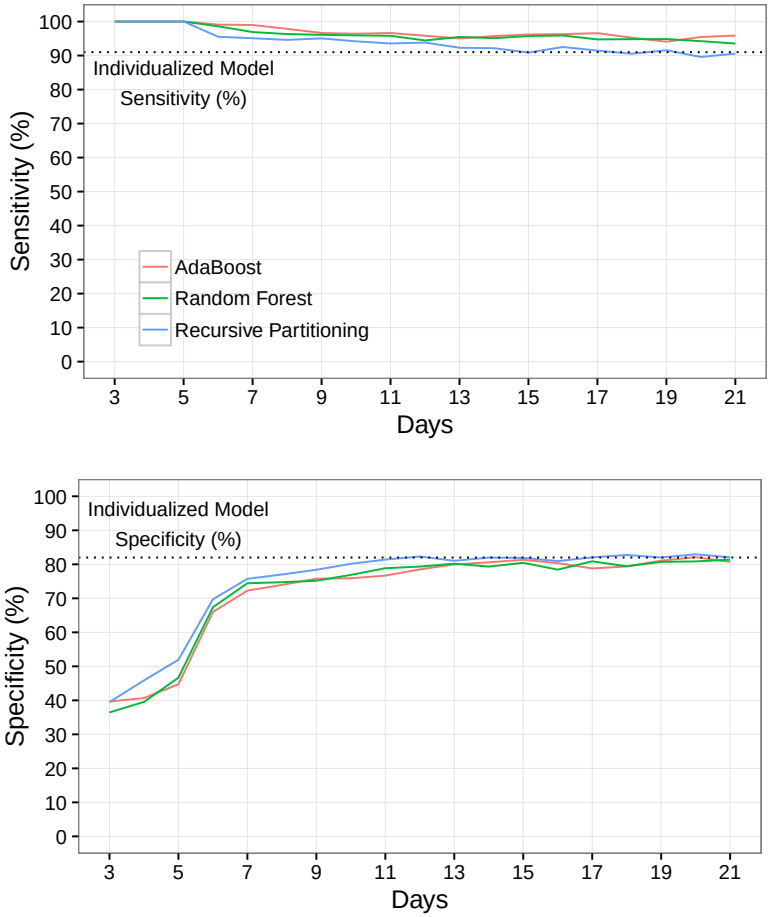


Figure 6.6: Prediction results for the online learning approach.

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

6.5 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, as discussed in Chapter 4 this approach has some limitations when used on mobile phones *in-the-wild*. Therefore, we evaluate our models by adopting an online learning approach. We construct individualised models with the same three machine learning 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 6.6 presents the prediction accuracy of the models with the increasing number of days. As the best achievable accuracy of the individualised prediction model trained with the batch learning approach we use 82% specificity and 91% sensitivity (discussed in the previous section). The results show that in just seven days all models achieve a specificity of more than 70% and sensitivity of more than 95%.

6.6 Group-based Prediction

Users exhibit different behaviour in terms of mobile phone interaction. For this reason, the approach based on a generic model for training predictors (i.e., the approach based on a training set composed of data from all the users) fails to cope with such a diversity in users' behaviour. On the other hand, as shown in Figure 6.6, an individualised model takes around 13 days to be trained. This is the classic problem of *bootstrapping* prediction models, a typical issue that developers have to deal with in building anticipatory computing systems [PM15].

In order to address this issue, we construct and evaluate group-based prediction models. Here, each group is comprised of a set of users who share similar behavioural characteristics. This approach is inspired by the *Community Similarity Networks* method proposed by Lane et al. [LXL+11], which uses inter-person similarity measurements for training classifiers. It is worth noting that the metrics used for computing groups of users can be derived from the data that is available immediately after the installation of an app.

6.6.1 Measuring Similarity Between Users

In order to quantify similarity between users we rely on the following two types of inter-person similarity: notification interaction similarity and lifestyle similarity. Notification interaction similarity indicates the similarity between users' behaviour in terms of interaction with notifications. Lifestyle similarity enables us to measure the similarity in users with respect to their day-to-day physical activities.

To compute the *notification interaction similarity* value between a pair of users, we use six types of notification interaction-based characteristics:

(i) average seen time of notifications (i.e., time from the notification arrival until the time it was seen by the user);

(ii) average response time of notifications (i.e., time from the notification arrival until the time of its removal from the notification bar);

(iii) percentage of notifications handled on the mobile phone;

(iv) percentage of notifications clicked over the total number of notifications handled on the mobile phone;

(v) percentage of notifications seen on the phone but handled on a different device;

(vi) percentage of notifications that arrived when the phone is connected to the Internet via WiFi.

To compute the *lifestyle similarity* value between a pair of users, we use three types of daily activity-based characteristics:

(i) average time spent commuting by foot (includes commuting by bicycle) in a day;

(ii) average time spent commuting by vehicle in a day;

(iii) average time spent being still in a day.

As discussed earlier, we rely on the Google Activity Recognition API [GAR16] to obtain the user's physical activity information. The API classifies the activity as *walking*, *cycling*, *commuting in vehicle* or *still*. However, we merge *walking* and *cycling* together as *moving* since we assume that users interact only with their phone when they perform these activities rather than other devices such as laptop.

It is worth noting that our objective is to show the potential of optimising the similarity method for building group-based models. More refined models of similarity can be built, but this is outside the scope of this work.

We construct a vector for representing each user with the above nine characteristics (six for notification interaction similarity and three for lifestyle similarity) as the vector elements. The similarity between a pair of users is measured by computing the Mahalanobis distance [Mah36] between the two vectors. More formally, the similarity between users A and B is defined as:

$$sim(A, B) = \sqrt{(C_A - C_B)^\top S^{-1}(C_A - C_B)} \tag{6.1}$$

Here, C_A and C_B are characteristic vectors of the users A and B respectively. S is the covariance matrix computed by using the characteristic vectors of all the users.

6.6.2 Clustering User Groups

We use the hierarchical clustering algorithm to cluster users based on their similarity measures. A distance matrix is computed to represent the similarity of users between each other. This matrix is given as an input to the clustering model in order to build a hierarchy of the individual elements (i.e., users) that can be progressively merged to form clusters.

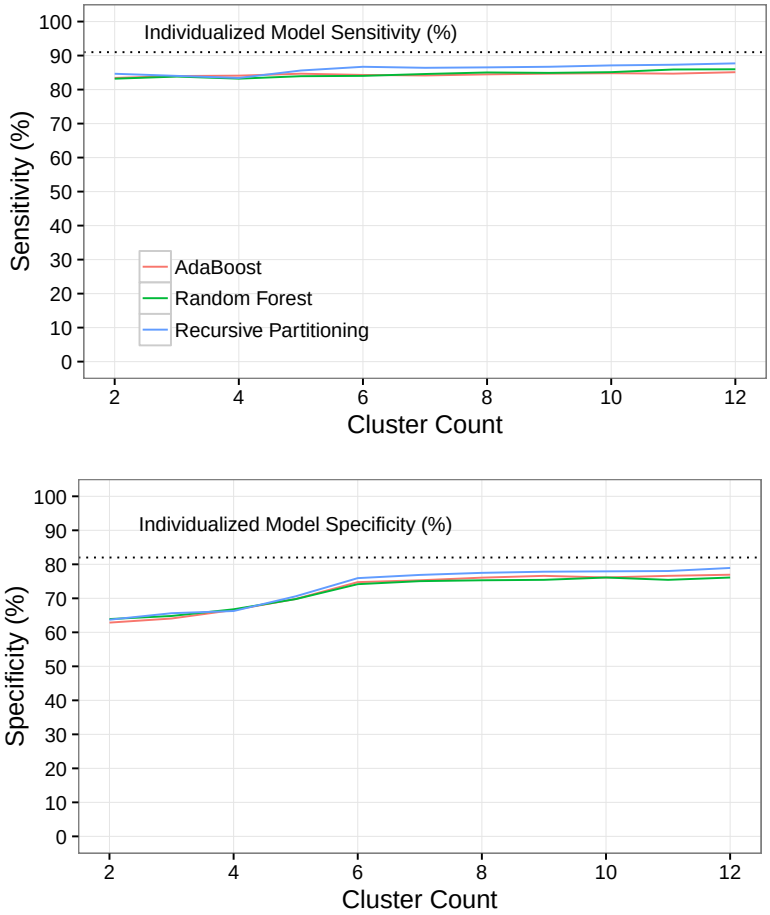


Figure 6.7: Prediction results for the group-based approach.

6.6.3 Constructing and Evaluating Group-based Models

Since the optimal number for groups (N) that should be used for constructing the group-based models is unknown, we set the value of $N \in [2, N\_max]$. We set N_max equal to 12 in order to avoid singletons (i.e., clusters with a single user). At the same time, the optimal value of $N \in [2, 12]$ does not guarantee that, by forming N clusters, no cluster will contain a single user. As the value of N increases, it is more probable that some clusters will consist of single users.

Now, we construct and evaluate the models for each group for predicting the device on which users of the relevant group will handle the given notifications. We use the k-fold cross validation approach with the value of k as 10 for evaluating group models.

In Figure 6.7 we present the prediction accuracy results in terms of sensitivity and specificity for the group-based models with $N \in [2, 12]$. The results show that the specificity increases significantly until N reaches the value of 6 and then the rate of increase become almost negligible for higher values of N. However, this increase is not reflected in a substantial change in sensitivity. Therefore, it is possible for example to divide the 24 users into 6 groups obtaining around 88% sensitivity and 75% specificity.

6.7 Limitations

The main limitation of this work resides in the fact that we do not have data for each device accessed by a user but for only two classes of device: personal phones and alternative devices (i.e., any device other than the phone). At the same time, the proposed model is highly generalisable 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 it is possible 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.

6.8 Summary

Notifications are increasingly adopted to attract user's attention on upcoming events and various pieces of information, such as new messages or social network updates. However, such notifications become disruptive and annoying when they are triggered on multiple devices. Current systems are oblivious of the user's preferences in terms of notification delivery, i.e., their preferences for receiving a specific piece of information on a specific device in a given situation. In order to address this problem, in this work we have proposed a novel solution for intelligent notification delivery in multi-device environments by analysing previous notification response behaviour 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 behaviour 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 generalised models trained using the data of all users and personalised ones trained using only the data of a single user, i.e., the owner of the device. We have shown that an individualised model is characterised by better prediction performance. More specifically, we have shown that it can predict the device on which the user will handle a notification in a given context with 82% specificity and 91% sensitivity. We have also discussed the implementation of an online predictor that is able to achieve a specificity of more than 70% and a sensitivity of 95% in just seven days.

Finally, we have shown that in the absence of previous training data, it is possible to bootstrap the models by using data of users that have similar behaviour patterns: this is achieved by clustering users based on notification interaction characteristics and lifestyle indicators extracted from the sensor data.

CHAPTER 7

CONCLUSIONS

Mobile phones have become a part of our everyday life. They provide always-on connectivity, high-speed data processing and advanced sensing. These affordances have made it a unique platform to push information in real-time and, thus, they represent a medium for receiving information in an effortless way. However, previous studies have shown that these inevitable notifications often arrive at inopportune moments, which can adversely affect the ongoing task [CCH01, CCH00a, MBDT02, BKC00] and affective state of the user [AB04, BK06]. This tension is exacerbated by the fact that individuals have to deal with a plethora of notifications in a day, some of which are disruptive [PCdO14] and also delivered on multiple devices at the same time by cross-platform applications. This fact makes these notifications potentially even more annoying.

The findings of previous studies, which are discussed in this thesis, provide evidence for the need a smart notification mechanism that can reduce the level of disruption by delivering the right information at the right time. Ideally, such a mechanism should not only aim at learning the observed user behaviour associated with the sensed contextual information and adapt the notification delivery process accordingly, but it should also possess the knowledge about *good behaviour* that can be exploited to improve the behaviour of users. For example, if a user reads the delivered emails on their mobile phone while driving, the notification mechanism should not learn this behaviour and deliver emails accordingly. Instead, the mechanism should infer that it is a *harmful behaviour*

to read notifications while driving and try to avoid sending unnecessary emails. Indeed, if the information is critical it should be delivered immediately regardless of the current situation. However, designing such an ideal notification delivery mechanism is extremely difficult and also it is out of the scope of this thesis. Therefore, this thesis focuses on reducing the level of disruption perceived by users every day by delivering information at opportune moments based on their actual behaviour.

At the same time, learning actual behaviour of users for interacting with notifications is extremely complex and depends on various contextual aspects both physical and cognitive. Indeed, the more information about the user's situation that the system can obtain, the better it will learn about the user's interruptibility. However, the technology is not so advanced yet to continuously monitor users' cognitive context without interacting with them. Such information can be gathered by relying on Experience Sampling Method (ESM) in which users are asked to register their cognitive context through the triggered questionnaires. However, it can only obtain their cognitive context in specific moments rather than as a continuous stream of data.

However, thanks to the sensing capabilities of mobile phones, we are able to capture users' physical contextual information. The knowledge about the recipient's notification-interaction logs and corresponding physical context provides a great opportunity to both monitor and learn users' behaviour for interacting with notifications. Therefore, intelligent systems can be designed for learning various aspects of the user's interruptibility. However, this thesis does not tackle the problem of designing an ideal mechanism for notifications delivery. Instead, this thesis is focused on designing a framework for intelligent notifications, which can: (i) understand the factors associated with users' receptivity to notifications and infer opportune moments for notification delivery; (ii) identify and hide or remove notifications that are not useful, or are uninteresting or irrelevant for the user; and (iii) forecast the best medium to deliver cross-platform notifications.

More generally, this thesis explores the use of mobile phone sensing and machine learning algorithms for designing an intelligent notification mechanism. We believe that

the contributions made by this thesis will enable developers of applications that leverage push notification mechanism to provide effective and interruptibility-aware services.

7.1 Summary of Contributions

In the following we summarise the key contributions of this thesis in relation to the research questions stated at the beginning of the dissertation:

Research Question 1: *What are the physical and cognitive contextual features that make people attentive and receptive to mobile notifications?*

Previous studies have shown that users' receptivity to a notification can be determined by: (i) their physical context; (ii) how interesting, entertaining, relevant and actionable its content is for them; (iii) the type of application that triggers it. However, there is still a lack of understanding concerning the cognitive factors influencing users' receptivity to mobile notifications. To bridge this gap, in Chapter 3 we presented two in-situ studies to investigate the impact of cognitive and notification design factors on users' perception, attentiveness and receptivity to mobile notifications. The contributions of this chapter are twofold. First, we have confirmed the validity of some past desktop interruptibility studies in a mobile setting. More specifically, we have shown that the ongoing task's type, complexity and completion level influences the disruption perceived from mobile notifications. Second, for the first time, we have investigated the role of notification presentation, sender-recipient relationship and emotional states for modelling interruptibility.

Research Question 2: *Can we predict the opportune moments to deliver notifications given a user's context?*

Until now, the focus of interruptibility management studies has been on the context in which a notification has been received and not on the actual content of the notification. In Chapter 4, we presented a study of mobile user interruptibility to investigate how users

behave when they receive specific types of content through mobile notifications arriving in different contexts and from different senders. Through an "in-the-wild" study, we found that users' receptivity to notifications varies with the information type. The communication notifications, where the sender is a family member or a relative of the user, have the highest acceptance rate. Moreover, we presented the design and implementation of a machine learning model that uses both content and context together for predicting the most opportune moment for the delivery of *in-the-wild* notifications carrying specific types of information. Finally, we show that a machine learning approach leads to more accurate predictions of users' interruptibility than alternative ones based on user-defined rules.

Research Question 3: *Can we infer users' preferences to receive notifications in different situations?*

As shown in the Chapter 4, users do not accept all notifications delivered to them because their receptivity relies on the content type and the sender of the message. Therefore, 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. In Chapter 5, we presented the design, implementation and evaluation of a novel interruptibility management solution that learns the types of information users prefer to receive via notifications in different situations based on the automatic extraction of rules by mining their interaction with mobile notifications. Another important contribution of this study is to design a "customisable system" rather than a "black-box" solution, which allows the system to improve its effectiveness by discarding rules that, according to users, should not be used for stopping notifications on their phones. Finally, we performed an "in-the-wild" evaluation of the system.

Research Question 4: *On which medium should a notification be delivered in multi-device environments?*

In the previous contributions, we conducted studies to infer opportune moments and to learn the types of notifications users prefer to receive in different contexts. However, today's notifications are triggered on multiple devices and the proposed mechanisms are oblivious to the user's preferences for receiving a specific piece of information on a specific device in a given situation. To address this challenge, in Chapter 6, we first investigated the relationship between users' behaviour in terms of handling notifications in a multi-device environment and several contextual dimensions. Then, we designed, implemented and evaluated individualised and generic models to predict the device on which the user will handle a notification in a given context. Finally, we have shown that user clustering techniques can be applied to build group-based predictors for initial model bootstrapping.

7.2 Discussion

In this thesis we envisioned a framework for push-based information delivery on mobile phones in an intelligent fashion in order to increase the impact of the delivered information without causing disruption for the users. The envisioned framework should not only aim to learn the observed user behaviour associated with the sensed contextual information. Instead, we believe that the intelligent information delivery framework should also keep a check on *harmful behaviour* of users.

However, the focus of this thesis is to contribute to the framework with several mechanisms for introducing intelligence in notifications by learning only the observed behaviour of users. More specifically, we presented mechanisms for predicting (i) opportune moments for delivering information, (ii) users' preferences for receiving different types of information in specific contexts, and (iii) the device on which users are likely to interact with the delivered information in their current context. We believe that this work contributes to the envisioned framework by investigating the use of notification content and users' context together for modelling users' notification interaction behaviour. However, there is a big scope of improvement in the framework by accessing the importance of

other physical, social and cognitive factors for modelling users' notification interaction behaviour.

Some lessons learnt from the studies presented in this thesis can be summarised as follows:

- the notification content plays an important role for making a notification disruptive.
- there is no difference in learning users' notification interaction behaviour through different machine learning algorithms.
- notification interaction behaviour varies among users. In other words, it is very difficult to model such a behaviour through a generic model for all.
- it is very important to make the predictive model transparent to users. This not just allows the system to gain users' trust but also it enable the system to improve itself by learning from the incorrect predictions.
- a model does not take long to train and adapt itself by using an online learning approach.

7.3 Future Directions

This thesis has addressed various challenges concerning the design of a framework for intelligent mobile notifications. However, it has also raised some interesting questions that need further investigations, which might be considered as additional key aspects for realising the vision of building intelligent mechanisms that are able to deliver the *right* information in the *right* context.

Deferring Notifications. One of the key question that needs to be investigates is the following: should we defer a notification if it is not an opportune moment and for how long? Until now, interruptibility management studies have focused on inferring if the

current moment is opportune to deliver notifications or not. We believe that in order to be more effective, an interruptibility management system should not only predict if the current moment is opportune to send a notification, but in case the current moment is not the right one it should also anticipate the best moment in the nearest future. This would enable the overlying application to decide whether the notification should be deferred until the predicted opportune moment or not.

Previous studies have proposed mechanisms for anticipating certain types of context modalities such as location. We believe that similar prediction models can be adopted or developed in order to address this challenge.

Monitoring Cognitive Context. The state-of-the-art approaches for interruptibility management rely the analysis of users' physical context. However, as discussed in this thesis, users' interruptibility might also be associated with their cognitive context. There is indeed a need for developing and evaluating mechanisms for automatically capturing the level of users' engagement with the current task, difficulty of the interrupting task, and similar cognitive factors that might influence interruptibility. One of the potential approaches might be to explore the use of affective computing [PP97] in order to monitor users' emotional states.

Modelling for Multiple Devices. This thesis has examined the use of mobile sensing to model user behaviour for interacting with cross-platform notifications. However, our study was limited to predicting only two classes of devices: personal phones and alternative devices (i.e., any device other than the phone). This work can be extended to a variety of devices, such as phones, wearables, tablets and desktops. There are challenges related to heterogeneity of these platforms, which might be tackled by developing a thin layer of middleware on a set of libraries.

Large-scale Study. Similarly to previous interruptibility management projects, the studies presented in this thesis are performed with a small number of participants and over a short duration of time. At the same time, these studies are publicised through the network of the researchers performing the studies. This could introduce the bias of the self-selected sample of users and thus the behaviour of a certain group of user (within a network) might be different from others. Moreover, almost all previous studies are based on a first phase of data collection and then on a second phase focused on the development and validation of prediction models, which is usually performed offline.

Therefore, the results presented in these studies do not have ecological validity as the collected datasets might be biased towards a certain group of population. We believe that in order to further validate the robustness of the interruptibility prediction models and their ecological validity, there is a need to perform large scale study as well as *in-the-wild* evaluations. We also believe that reproducing these studies is essential in different social context and users' demographics.

LIST OF REFERENCES

[AB04] Piotr D Adamczyk and Brian P Bailey. If not now, when?: the effects of interruption at different moments within task execution. In *CHI'04*, pages 271–278. ACM, April 2004.

[AGKO04] Lauri Aalto, Nicklas Göthlin, Jani Korhonen, and Timo Ojala. Bluetooth and WAP push based location-aware mobile advertising system. In *MobiSys'04*, pages 49–58. ACM, June 2004.

[AIB05] Piotr D Adamczyk, Shamsi T Iqbal, and Brian P Bailey. A method, system, and tools for intelligent interruption management. In *Workshop on Task models and diagrams*, pages 123–126. ACM, September 2005.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD'93*, pages 207–216. ACM, June 1993.

[And16a] Android's Notification Listener Service, 2016. `http://developer.android.com/reference/android/service/notification/NotificationListenerService.html` [Retrieved on 01-12-2016].

[And16b] *Heads-up Notifications*, 2016. `http://developer.android.com/guide/topics/ui/notifiers/notifications.html\#Heads-up` [Retrieved on 01-12-2016].

[Atk53] John W Atkinson. The achievement motive and recall of interrupted and completed tasks. *Journal of Experimental Psychology*, 46(6):381–390, 1953.

[Bad76] Alan D Baddeley. *The psychology of memory*. Basic Books, 1976.

[Bea82] Jackson Beatty. Task-evoked pupillary responses, processing load, and the structure of processing resources. *Psychological bulletin*, 91(2):276–292, 1982.

[BEG98] Mark B. Edwards and Scott D Gronlund. Task interruption and its effects on memory. *Memory*, 6(6):665–687, 1998.

[BK06] Brian P Bailey and Joseph A Konstan. 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):685–708, 2006.

[BK10] Michael W Berry and Jacob Kogan. *Text mining: applications and theory*. John Wiley & Sons, 2010.

[BKC00] Brian P Bailey, Joseph A Konstan, and John V Carlis. Measuring the effects of interruptions on task performance in the user interface. In *SMC'00*, pages 757–762. IEEE, October 2000.

[BKC01] Brian P Bailey, Joseph A Konstan, and John V Carlis. The effects of interruptions on task performance, annoyance, and anxiety in the user interface. In *INTERACT'01*, pages 593–601. IEEE, July 2001.

[BMT04] J. B. Begole, Nicholas E Matsakis, and John C Tang. Lilsys: sensing unavailability. In *CSCW'04*, pages 511–514. ACM, November 2004.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[BW86] Rolf Braune and Christopher D Wickens. Time-sharing revisited: Test of a componential model for the assessment of individual differences. *Ergonomics*, 29(11):1399–1414, 1986.

[CCH00a] Mary Czerwinski, Edward Cutrell, and Eric Horvitz. Instant messaging and interruption: Influence of task type on performance. In *OZCHI'00*, pages 361–367. ACM, November 2000.

[CCH00b] Mary Czerwinski, Edward Cutrell, and Eric Horvitz. Instant messaging: Effects of relevance and timing. *People and computers XIV: Proceedings of HCI*, 2:71–76, 2000.

[CCH01] Edward Cutrell, Mary Czerwinski, and Eric Horvitz. Notification, disruption, and memory: Effects of messaging interruptions on memory and performance. In *Interact'01*, pages 263–269. IOS Press, July 2001.

[CCM90] Ph Cabon, ALEXM Coblentz, and REGIS Mollard. Interruption of a monotonous activity with complex tasks: effects of individual differences. *Human Factors and Ergonomics Society Annual Meeting*, 34(13):912–916, 1990.

[CL83] Mihaly Csikszentmihalyi and Reed Larson. The experience sampling method. *New Directions for Methodology of Social and Behavioral Science*, 15:41–56, 1983.

[CL14] Mihaly Csikszentmihalyi and Reed Larson. Validity and reliability of the experience-sampling method. In *Flow and the Foundations of Positive Psychology*, pages 35–54. Springer, 2014.

[Cla96] Herbert H Clark. *Using language*. Cambridge University Press, 1996.

[CLO16] CLOC – Count Lines of Code, 2016. `http://cloc.sourceforge.net` [Retrieved on 01-12-2016].

[CM15] Luca Canzian and Mirco Musolesi. Trajectories of depression: unobtrusive monitoring of depressive states by means of smartphone mobility traces analysis. In *UbiComp'15*, pages 1293–1304. ACM, September 2015.

[CMT+08] Sunny Consolvo, David W McDonald, Tammy Toscos, Mike Y Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, et al. Activity Sensing in the Wild: a Field Trial of UbiFit Garden. In *CHI'08*, pages 1797–1806. ACM, April 2008.

[CNM83] Stuart K Card, Allen Newell, and Thomas P Moran. *The psychology of human-computer interaction*. L. Erlbaum Associates Inc., 1983.

[CS89] Herbert H Clark and Edward F Schaefer. Contributing to discourse. *Cognitive science*, 13(2):259–294, 1989.

[CT15] Yung-Ju Chang and John C Tang. Investigating mobile users' ringer mode usage and attentiveness and responsiveness to communication. In *MobileHCI'15*, pages 6–15. ACM, August 2015.

[DDM16] Android DDMS, 2016. `http://developer.android.com/tools/debugging/ddms.html` [Retrieved on 01-12-2016].

[Dey01] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[DFAB93] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-Computer Interaction*. Prentice Hall, 1993.

[DJA93] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. Wizard of oz studieswhy and how. *Knowledge-based systems*, 6(4):258–266, 1993.

[DP15] Tilman Dingler and Martin Pielot. I'll be there for you: Quantifying attentiveness towards mobile messaging. In *MobileHCI'15*, pages 1–5. ACM, September 2015.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD'96*, pages 226–231. ACM, August 1996.

[EP09] Nathan Eagle and Alex Sandy Pentland. Eigenbehaviors: Identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, 2009.

[FEW12] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *SPSM'12*, pages 33–44. ACM, October 2012.

[FGB11] Joel E Fischer, Chris Greenhalgh, and Steve Benford. Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications. In *MobileHCI'11*, pages 181–190. ACM, September 2011.

[FHA+05] James Fogarty, Scott E. Hudson, Christopher G. Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny C. Lee, and Jie Yang. Predicting Human Interruptibility with Sensors. *ACM Transactions on Computer-Human Interaction*, 12(1):119–146, 2005.

[FHL04] James Fogarty, Scott E Hudson, and Jennifer Lai. Examining the robustness of sensor-based statistical models of human interruptibility. In *CHI'04*, pages 207–214. ACM, April 2004.

[FKA+05] James Fogarty, Andrew J Ko, Htet Htet Aung, Elspeth Golden, Karen P Tang, and Scott E Hudson. Examining task engagement in sensor-based

statistical models of human interruptibility. In *CHI'05*, pages 331–340. ACM, April 2005.

[FMK+10] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In *MobiSys'10*, pages 179–194. ACM, June 2010.

[Fri76] Jerome H Friedman. A recursive partitioning decision rule for nonparametric classification. *IEEE Transactions on Computers*, 26(4):404–408, 1976.

[FS10] Marcus Foth and Ronald Schroeter. Enhancing the experience of public transport users with urban screens and mobile applications. In *MindTrek'10*, pages 33–40. ACM, October 2010.

[FY38] Ronald Aylmer Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research*. Longman, 1938.

[FYB+10] Joel E Fischer, Nick Yee, Victoria Bellotti, Nathan Good, Steve Benford, and Chris Greenhalgh. Effects of content and time of delivery on receptivity to mobile interruptions. In *MobileHCI'10*, pages 103–112. ACM, September 2010.

[GAR16] Google's Activity Recognition Application. `http://developer.android.com/training/location/activity-recognition.html` [Retrieved on 01-12-2016], 2016.

[GB89] Tony Gillie and Donald Broadbent. What makes interruptions disruptive? a study of length, similarity, and complexity. *Psychological research*, 50(4):243–250, 1989.

[GJ09] Sukeshini A Grandhi and Quentin Jones. Conceptualizing interpersonal interruption management: A theoretical framework and research program. In *HICSS'09*, pages 1–10. IEEE, January 2009.

[GS86] Barbara J Grosz and Candace L Sidner. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204, 1986.

[HAS05] Eric Horvitz, Johnson Apacible, and Muru Subramani. Balancing awareness and interruption: Investigation of notification deferral policies. In *UM'05*, pages 433–437. 2005.

[HFA+03] Scott Hudson, James Fogarty, Christopher Atkeson, Daniel Avrahami, Jodi Forlizzi, Sara Kiesler, Johnny Lee, and Jie Yang. Predicting human interruptibility with sensors: a wizard of oz feasibility study. In *CHI'03*, pages 257–264. ACM, April 2003.

[HFH+09] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[HI05] Joyce Ho and Stephen S. Intille. Using Context-Aware Computing to Reduce the Perceived Burden of Interruptions from Mobile Devices. In *CHI'05*, pages 909–918. ACM, April 2005.

[HJH99] Eric Horvitz, Andy Jacobs, and David Hovel. Attention-sensitive alerting. In *UAI'99*, pages 305–313. Morgan Kaufmann Publishers Inc., August 1999.

[HKA04] Eric Horvitz, Paul Koch, and Johnson Apacible. BusyBody: Creating and Fielding Personalized Models of the Cost Interruption. In *CSCW'04*, November 2004.

[HKPH03] Eric Horvitz, Carl Kadie, Tim Paek, and David Hovel. Models of attention in computing and communication: from principles to applications. *Communications of the ACM*, 46(3):52–59, 2003.

[HKS+05] Eric Horvitz, Paul Koch, Raman Sarin, Johnson Apacible, and Muru Subramani. Bayesphone: Precomputation of context-sensitive policies for inquiry and action in mobile devices. In *User Modeling'05*, pages 251–260. Springer, July 2005.

[HL93] Bert Hoeks and Willem JM Levelt. Pupillary dilation as a measure of attention: A quantitative system analysis. *Behavior Research Methods, Instruments, & Computers*, 25(1):16–26, 1993.

[HS88] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183, 1988.

[Hus87] MG Husain. Immediate and delayed recall of completed-interrupted tasks by high and low anxious subjects. *Manas*, 34(2):67–71, 1987.

[IAZB05] Shamsi T Iqbal, Piotr D Adamczyk, Xianjun Sam Zheng, and Brian P Bailey. Towards an index of opportunity: understanding changes in mental workload during task execution. In *CHI'05*, pages 311–320. ACM, April 2005.

[IB06] Shamsi T Iqbal and Brian P Bailey. Leveraging characteristics of task structure to predict the cost of interruption. In *CHI'06*, pages 741–750. ACM, April 2006.

[IB07] Shamsi T Iqbal and Brian P Bailey. Understanding and developing models for detecting and differentiating breakpoints during interactive tasks. In *CHI'07*, pages 697–706. ACM, April 2007.

[IB10] Shamsi T Iqbal and Brian P Bailey. Oasis: A framework for linking notification delivery to the perceptual structure of goal-directed tasks. *ACM Transactions on Computer-Human Interaction*, 17(4):15, 2010.

[IH10] Shamsi T Iqbal and Eric Horvitz. Notifications and awareness: a field study of alert usage and preferences. In *CSCW'10*, pages 27–30. ACM, February 2010.

[Int15] International Telecommunication Union. Measuring the Information Society. Technical report, 2015. `http://www.itu.int/en/ITU-D/Statistics/Documents/publications/misr2015/MISR2015-w5.pdf` [Retrieved on 01-12-2016].

[IZB04] Shamsi T Iqbal, Xianjun Sam Zheng, and Brian P Bailey. Task-evoked pupillary response to mental workload in human-computer interaction. In *CHI'04*, pages 1477–1480. ACM, April 2004.

[JH98] Susan Joslyn and Earl Hunt. Evaluating individual differences in response to time-pressure situations. *Journal of Experimental Psychology: Applied*, 4(1):16–43, 1998.

[Joa02] Thorsten Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms*. Kluwer Academic Publishers, 2002.

[KH08] Ashish Kapoor and Eric Horvitz. Experience sampling for building predictive user models: a comparative study. In *CHI'08*, pages 657–666. ACM, April 2008.

[KM81] John G Kreifeldt and ME McCarthy. *Interruption as a test of the user-computer interface*. 1981.

[KPD16] Kostadin Kushlev, Jason Proulx, and Elizabeth W Dunn. Silence your phones: Smartphone notifications increase inattention and hyperactivity symptoms. In *CHI'16*, pages 1011–1020. ACM, April 2016.

[Kru16] John Krumm. *Ubiquitous computing fundamentals*. CRC Press, 2016.

[Lat98] Kara A Latorella. Effects of modality on interrupted flight deck performance: Implications for data link. In *HFES'98*, pages 87–91. SAGE, October 1998.

[Lat99] Kara A Latorella. *Investigating interruptions: Implications for flightdeck performance*. 1999.

[LC94] David D Lewis and Jason Catlett. Heterogeneous uncertainty sampling for supervised learning. In *ICML'94*, pages 148–156. ACM, June 1994.

[LCC+11] Nicholas D. Lane, Tanzeem Choudhury, Andrew Campbell, Mashfiqui Mohammod, Mu Lin, Xiaochao Yang, Afsaneh Doryab, Hong Lu, Shahid Ali, and Ethan Berke. BeWell: A Smartphone Application to Monitor, Model and Promote Wellbeing. In *Pervasive Health'11*, pages 23–26. ACM, May 2011.

[LCLP12] Hosub Lee, Young Sang Choi, Sunjae Lee, and IP Park. Towards unobtrusive emotion recognition for affective social communication. In *CCNC'12*, pages 260–264. ACM, January 2012.

[LFN15] Kyungmin Lee, Jason Flinn, and Brian Noble. The case for operating system management of user attention. In *HotMobile'15*, pages 87–91. ACM, February 2015.

[LLLZ13] Robert LiKamWa, Yunxin Liu, Nicholas D Lane, and Lin Zhong. Moodscope: Building a mood sensor from smartphone usage patterns. In *MobiSys'13*, pages 389–402. ACM, June 2013.

[LML+10] Nicholas D. Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T. Campbell. A Survey of Mobile Phone Sensing. *IEEE Communications Magazine*, 48(9):140–150, 2010.

[LMRF+12] Hong Lu, Gokul T. Chittaranjan Mashfiqui Rabbi, Denise Frauendorfer, Marianne Schmid Mast, Andrew T. Campbell, Daniel Gatica-Perez, and Tanzeem Choudhury. StressSense: Detecting Stress in Unconstrained Acoustic Environments using Smartphones. In *UbiComp'12*, pages 351–360. ACM, September 2012.

[LPL+09] Hong Lu, Wei Pan, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. SoundSense: Scalable Sound Sensing for People-centric Applications on Mobile Phones. In *MobiSys'09*, pages 165–178. ACM, June 2009.

[LPR+13] Neal Lathia, Veljko Pejovic, Kiran K. Rachuri, Cecilia Mascolo, Mirco Musolesi, and Peter J. Rentfrow. Smartphones for Large-Scale Behaviour Change Intervention. *IEEE Pervasive Computing*, 12(12):66–73, July 2013.

[LRMR13] Neal Lathia, Kiran K. Rachuri, Cecilia Mascolo, and George Roussos. Open Source Smartphone Libraries for Computational Social Science. In *MCSS'13*, pages 911–920. ACM, September 2013.

[LTCD15] Hugo Lopez-Tovar, Andreas Charalambous, and John Dowell. Managing smartphone interruptions through adaptive modes and modulation of notifications. In *IUI'15*, pages 996–999. ACM, March 2015.

[LXL+11] Nicholas D. Lane, Ye Xu, Hong Lu, Shaohan Hu, Tanzeem Choudhury, Andrew T. Campbell, and Feng Zhao. Enabling Large-scale Human Activity Inference on Smartphones using Community Similarity Networks (CSN). In *UbiComp'11*, pages 355–364. ACM, September 2011.

[Mah36] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

[MBDT02] Christopher A Monk, Deborah A Boehm-Davis, and J Gregory Trafton. The attentional costs of interrupting task performance at various stages. In *HFES'02*, pages 1824–1828. SAGE, October 2002.

[McF97] Daniel C McFarlane. Interruption of people in human-computer interaction: A general unifying definition of human interruption and taxonomy. Technical report, DTIC Document, 1997.

[MCSN03] D Scott McCrickard, Christa M Chewar, Jacob P Somervell, and Ali Ndiwalana. A model for notification systems evaluationassessing user goals for

multitasking activity. *ACM Transactions on Computer-Human Interaction*, 10(4):312–338, 2003.

[MHM16a] Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. PrefMiner: Mining User's Preferences for Intelligent Mobile Notification Management. In *Proceedings of UbiComp'16*, pages 1223–1234, Heidelberg, Germany, September 2016. ACM.

[MHM16b] Abhinav Mehrotra, Robert Hendley, and Mirco Musolesi. Towards Multimodal Anticipatory Monitoring of Depressive States through the Analysis of Human-Smartphone. In *Adjunct UbiComp'16*, pages 1132–1138, Heidelberg, Germany, September 2016. ACM.

[ML02] Daniel C McFarlane and Kara A Latorella. The scope and importance of human interruption in human-computer interaction design. *Human-Computer Interaction*, 17(1):1–61, 2002.

[MMHP15] Abhinav Mehrotra, Mirco Musolesi, Robert Hendley, and Veljko Pejovic. Designing Content-driven Intelligent Notification Mechanisms for Mobile Applications. In *UbiComp'15*, pages 813–824. ACM, September 2015.

[MN86] Yoshiro Miyata and Donald A Norman. Psychological issues in support of multiple activities. *User centered system design: New perspectives on human-computer interaction*, 1986.

[MN+98] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI'98*, pages 41–48. MIT Press, July 1998.

[MPM14] Abhinav Mehrotra, Veljko Pejovic, and Mirco Musolesi. SenSocial: A Middleware for Integrating Online Social Networks and Mobile Sensing Data Streams. In *Middleware'14*, pages 205–216. ACM, December 2014.

[MPV+16] Abhinav Mehrotra, Veljko Pejovic, Jo Vermeulen, Robert Hendley, and Mirco Musolesi. My Phone and Me: Understanding People's Receptivity to Mobile Notifications. In *CHI'16*, pages 1021–1032. ACM, April 2016.

[MVPM15] Abhinav Mehrotra, Jo Vermeulen, Veljko Pejovic, and Mirco Musolesi. Ask, But Don't Interrupt: The Case for Interruptibility-Aware Mobile Experience Sampling. In *UbiComp'15 Adjunct*, pages 723–732. ACM, September 2015.

[NTS02] Minoru Nakayama, Koji Takahashi, and Yasutaka Shimizu. The act of task difficulty and eye-movement frequency for the'oculo-motor indices'. In *ETRA'02*, pages 37–42. ACM, March 2002.

[ONN+16] Tadashi Okoshi, Hiroki Nozaki, Jin Nakazawa, Hideyuki Tokuda, Julian Ramos, and Anind K Dey. Towards attention-aware adaptive notification on smart phones. *Pervasive and Mobile Computing*, 26:17–34, 2016.

[Ord06] Carlos Ordonez. Comparing association rules and decision trees for disease prediction. In *HIKM'06*, pages 17–24. ACM, November 2006.

[ORMR12] Antti Oulasvirta, Tye Rattenbury, Lingyi Ma, and Eeva Raita. Habits make smartphone use more pervasive. *Personal and Ubiquitous Computing*, 16(1):105–114, 2012.

[ORN+15] Tadashi Okoshi, Julian Ramos, Hiroki Nozaki, Jin Nakazawa, Anind K Dey, and Hideyuki Tokuda. Reducing users' perceived mental effort due to interruptive notifications in multi-device mobile environments. In *UbiComp'15*, pages 475–486. ACM, September 2015.

[PCdO14] Martin Pielot, Karen Church, and Rodrigo de Oliveira. An in-situ study of mobile phone notifications. In *MobileHCI'14*, pages 233–242. ACM, September 2014.

[PdOKO14] Martin Pielot, Rodrigo de Oliveira, Haewoon Kwak, and Nuria Oliver. Didn't you see my message?: predicting attentiveness to mobile instant messages. In *CHI'14*, pages 3319–3328. ACM, April 2014.

[PDPO15] Martin Pielot, Tilman Dingler, Jose San Pedro, and Nuria Oliver. When attention is not scarce-detecting boredom from mobile phone usage. In *UbiComp'15*, pages 825–836. ACM, September 2015.

[Pea00] Karl Pearson. X. 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):157–175, 1900.

[Pie14] Martin Pielot. Large-scale evaluation of call-availability prediction. In *UbiComp'14*, pages 933–937. ACM, September 2014.

[PM75] Karl H Pribram and Diane McGuinness. Arousal, activation, and effort in the control of attention. *Psychological Review*, 82(2):116–149, 1975.

[PM14a] Veljko Pejovic and Mirco Musolesi. Anticipatory mobile computing for behaviour change interventions. In *UbiComp'14 Adjunct*, pages 17–24, Seattle, WA, USA, September 2014. ACM.

[PM14b] Veljko Pejovic and Mirco Musolesi. Interruptme: designing intelligent prompting mechanisms for pervasive applications. In *UbiComp'14*, pages 897–908. ACM, September 2014.

[PM15] Veljko Pejovic and Mirco Musolesi. Anticipatory mobile computing: A survey of the state of the art and research challenges. *ACM Computing Surveys*, 47(3):1–47, 2015.

[PMM15] Veljko Pejovic, Abhinav Mehrotra, and Mirco Musolesi. Investigating The Role of Task Engagement in Mobile Interruptibility. In *MobileHCI'15 Adjunct*, pages 1100–1105, Copenhagen, Denmark, August 2015. ACM.

[PP97] Rosalind W Picard and Roalind Picard. *Affective computing*. MIT press Cambridge, 1997.

[qda16] qdapDictionaries for R, 2016. `http://trinker.github.io/qdapDictionaries/` [Retrieved on 01-12-2016].

[RDV11] Stephanie Rosenthal, Anind K Dey, and Manuela Veloso. Using decision-theoretic experience sampling to build personalized mobile phone interruption models. In *PerCom'11'*, pages 170–187. ACM, March 2011.

[RMM+10] Kiran K. Rachuri, Mirco Musolesi, Cecilia Mascolo, Jason Rentfrow, Chris Longworth, and Andrius Aucinas. EmotionSense: A Mobile Phones based Adaptive Platform for Experimental Social Psychology Research. In *UbiComp'10*, pages 281–290. ACM, September 2010.

[ROM01] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[RS04] Laura Elena Raileanu and Kilian Stoffel. Theoretical comparison between the gini index and information gain criteria. *Annals of Mathematics and Artificial Intelligence*, 41(1):77–93, 2004.

[SCC02] William R Shadish, Thomas D Cook, and Donald Thomas Campbell. *Experimental and quasi-experimental designs for generalized causal inference*. Houghton, Mifflin and Company, 2002.

[Sel56] Hans Selye. *The stress of life*. McGraw-Hill, 1956.

[Sel04] Martin EP Seligman. Can happiness be taught? *Daedalus*, 133(2):80–87, 2004.

[SH77] Philip H Swain and Hans Hauska. The decision tree classifier: Design and potential. *IEEE Transactions on Geoscience Electronics*, 15(3):142–147, 1977.

[Sno16] Snowball Stemming Libraries for R, 2016. `http://snowball.tartarus.org` [Retrieved on 01-12-2016].

[Spe16] *Android's SpeechRecognizer API*, 2016. `http://developer.android.com/reference/android/speech/SpeechRecognizer.html` [Retrieved on 01-12-2016].

[SSHD+14] Alireza Sahami Shirazi, Niels Henze, Tilman Dingler, Martin Pielot, Dominik Weber, and Albrecht Schmidt. Large-scale assessment of mobile notifications. In *CHI'14*, pages 3055–3064. ACM, April 2014.

[SSJ74] Harvey Sacks, Emanuel A Schegloff, and Gail Jefferson. A simplest systematics for the organization of turn-taking for conversation. *Language*, 1974.

[SVV99] Cheri Speier, Joseph S Valacich, and Iris Vessey. The influence of task interruption on individual decision making: An information overload perspective. *Decision Sciences*, 30(2):337–360, 1999.

[SVV03] Cheri Speier, Iris Vessey, and Joseph S Valacich. The effects of interruptions, task complexity, and information presentation on computer-supported decision-making performance. *Decision Sciences*, 34(4):771–797, 2003.

[Tha89] Robert E Thayer. *The biopsychology of mood and arousal*. Oxford University Press, 1989.

[TM15] Fani Tsapeli and Mirco Musolesi. Investigating causality in human behavior from smartphone sensor data: a quasi-experimental approach. *EPJ Data Science*, 4(1):1–15, 2015.

[WB97] Mark Weiser and John Seely Brown. *The coming age of calm technology*. Springer, 1997.

[WCC+12] Tianyu Wang, Giuseppe Cardone, Antonio Corradi, Lorenzo Torresani, and Andrew T Campbell. WalkSafe: A Pedestrian Safety App for Mobile Phone Users Who Walk and Talk While Crossing Roads. In *HotMobile'12*, pages 5–14. ACM, February 2012.

[Wei65] Bernard Weiner. Need achievement and the resumption of incompleted tasks. *Journal of Personality and Social Psychology*, 1(2):165–168, 1965.

[Wei91] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.

[Wek16] Weka: Data Mining Software in Java, 2016. `http://www.cs.waikato.ac.nz/ml/weka/` [Retrieved on 01-12-2016].

[WMW15] Tilo Westermann, Sebastian Möller, and Ina Wechsung. Assessing the Relationship between Technical Affinity, Stress and Notifications on Smartphones. In *MobileHCI'15 Adjunct*, pages 652–659. ACM, September 2015.

[WVKH16] Dominik Weber, Alexandra Voit, Philipp Kratzer, and Niels Henze. In-situ investigation of notifications in multi-device environments. In *UbiComp'16*, pages 1259–1264. ACM, September 2016.

[XLL+13] Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell, and Tanzeem Choudhury. Preference, context and communities: A multi-faceted approach to predicting smartphone app usage patterns. In *ISWC'13*, pages 69–76. ACM, September 2013.

[YP97] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *ICML'97*, pages 412–420. ACM, July 1997.

[Zei27] Bluma Zeignaric. Das behalten erledigter und unerledigter handlungern. *Psychologische Forschung*, 9:1–85, 1927.

[ZOP+15] Darya L Zabelina, Daniel OLeary, Narun Pornpattananangkul, Robin Nusslock, and Mark Beeman. Creativity and sensory gating indexed by the p50: Selective versus leaky sensory gating in divergent thinkers and creative achievers. *Neuropsychologia*, 69:77–84, 2015.

[ZRLK99] Fred RH Zijlstra, Robert A Roe, Anna B Leonora, and Irene Krediet. Temporal factors in mental work: Effects of interrupted activities. *Journal of Occupational and Organizational Psychology*, 72(2):163–185, 1999.

[ZTQ+10] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Zhuoqing Morley Mao, and Lei Yang. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. In *CODES/ISSS'10*, pages 105–114. IEEE, October 2010.

CHAPTER 8

APPENDIX

Figure 8.1: **Screenshots for Android notification mechanism.**

8.1 Notification Presentation on Android Phones

In this section we demonstrate the mechanism of Android notification and how users can interact with notifications. As shown in Figure 8.1 presents the possible ways in which a notification can be viewed on an Android phone. In Figure 8.1(a) the notification is

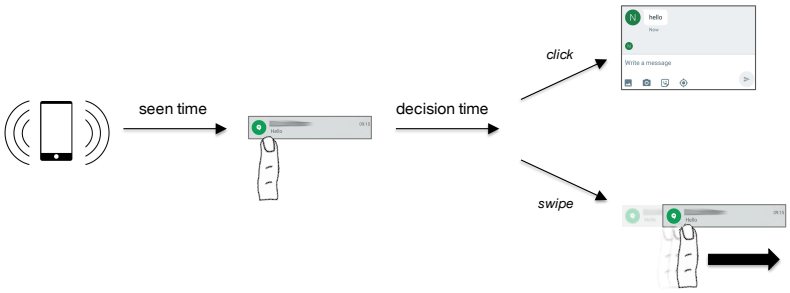


Figure 8.2: Ways of interacting with an Android notification.

presented on the phone's lock screen, which enables users to view and dismiss (by swiping sidewise) it without unlocking the the phone. However, if the user wants to interact with the notification they can click on it that launches the relevant app.

On the other hand, a user can also unlock the phone without interacting with the notifications. In that case, notification still appears on the notification bar as shown in Figure 8.1(b). To interact with these notifications now the user can pull the notification bar down as shown in Figure 8.1(c).

8.2 Users' Interaction with a Notification

As shown in Figure 8.2, a user can view a notification and then decide how to handle it. This enables us to capture three time measurements for each notification: the time of notification arrival (a), the time when the notification is seen (b), and the time when the user accepted (c1) or dismissed (c2) the notification. It is worth noting that in order to accept a notification (c1) the user has to click on it which launches the relevant app, whereas for dismissing the notification the user can simply swipe it sidewise.