TÉCNICO LISBOA

Active Data Collection of Health Data in Mobile Devices

Ana Rita Bamansá Siles Machado

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Alexandre José Malheiro Bernardino
Prof. Plinio Moreno López

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Alexandre José Malheiro Bernardino
Member of the Committee: Prof. Athanasios Vourvopoulos

December 2021

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

I would like to start by thanking my supervisors Alexandre Bernardino, Plinio López, and Heitor Cardoso, for their time, support, and sharing of knowledge and valuable insights. This work would also not have been possible without the constant love and support of my parents and sisters, who have always believed in me more than I ever will. I will never be able to fully express how thankful I am, but I promise to never stop trying.

To all my friends, even those who are not today such a big part of my life as they have been before, thank you. For the past 7 years, through the highs and the lows, you have been there. From dinners and nights of drinking to nights of studying and working on projects, I would not have made it through college without you. In different ways, all of you have helped me grow and shaped me into the supposed adult I am today. Nonetheless, some of you have had a major role in that process: Bernardo, because no matter how much I try, I could never erase your presence and impact in my life, and despite everything, through the good and the really really bad moments, you have taught me more about myself than anyone, or anything else, ever has. Joana, Mafalda, and Sara, the ”OG” group I know I can always count on no matter how many years go by. To Sara, a very special thank you, your daily support and pep talks have kept me going through much more than I ever thought I could handle. I cannot begin to express how big of an impact you have made in my life, so I will not try, but once again, thank you. To Sofia, I suppose it takes two slightly unhinged people to understand each other as we do. Thank you for all the support and for being there for me, even in those moments when you probably did not really want to. Your friendship and many cute dog pictures have made my days consistently better. Finally, to Joana and Joana, who have been with me for many more than 7 years now but with whom, no matter how much time goes by, it always seems like we never stopped seeing each other every day. To you all, I thank you for all the, not always pleasant, lessons you have taught me.

Lastly, I would like to thank my co-workers (and friends), from whom I could have never expected to receive this overwhelming amount of support. You have asked and actually listened to my problems, questions, and countless times just complaints, helping me push through many of the more challenging moments of the past year. Your presence, patience, computational power, and overall willingness to help wherever possible made this thesis possible, and for that, I would like to express my deepest gratitude.

To each and every one of you, thank you.

Abstract

This thesis aims to develop an intelligent notification system to help sustain user engagement in mHealth applications that support self-management. We consider a mHealth application installed in the user's smartphone that periodically requests data about their state and activity. The proposed approach applies Reinforcement Learning (RL) to learn what moments are the most opportune to send notifications, using easily obtainable non-sensitive data (such as the time or day of the week) and application usage history. This history allows the RL agent to remember how the user has reacted to notifications in the past. Various formulations of the problem are designed, developing several alternatives for the components required by the implemented RL algorithms. Through the permutation of these components, the most desirable combination of initialization method, algorithm, state, and reward, is searched. A simulator is developed to mimic the behavior of a typical user and utilized to test the developed combinations with users experiencing distinct lifestyles. This work shows promising results and, although requiring further testing to be fully validated, demonstrates that an efficient and well-balanced notification system can be built with simple formulations of an RL problem and algorithm. Furthermore, it proves that there is no absolute necessity for this type of system to access sensitive data such as the users' location or surrounding noise. This approach diminishes privacy issues that might concern the user and limits sensor and hardware concerns, such as lapses in collected data or battery drainage, by not requiring any information provided by the device's sensors.

Keywords

mHealth; Notifications; Machine Learning; Personalization; Reinforcement Learning; Receptivity.

Resumo

Nesta tese pretendemos desenvolver um sistema de notificações inteligente, destinado a aplicações de mHealth, que melhore o compromisso a longo prazo dos utilizadores. Considera-se uma aplicação instalada no telemóvel do utilizador, que pede periodicamente informação acerca do seu estado e atividade. A metodologia proposta utiliza Aprendizagem por Reforço, permitindo a aprendizagem de quais os momentos mais oportunos para enviar notificações. Tal ocorre com base em informação facilmente acessível e que não é privada. Primariamente, utiliza-se o histórico de reacções, isto é, como o utilizador reagiu no passado às notificações recebidas. São desenhadas várias formulações deste problema, desenvolvendo variantes de cada um dos componentes que os algoritmos implementados requerem. Estes componentes são posteriormente permutados de modo a encontrar a melhor combinação de método de inicialização, algoritmo, estado e recompensa. Desenvolve-se também um simulador para reflectir o comportamento dos utilizadores. Este é utilizado para testar as combinações criadas, replicando o dia-a-dia de vários utilizadores com estilos de vida distintos. Este trabalho exibe resultados promissores e, apesar de necessitar de uma maior variedade de testes para ser totalmente validado, demonstra que um sistema eficiente e equilibrado pode ser construído usando formulações simples de um problema de Aprendizagem por Reforço. Adicionalmente, comprova que não existe a obrigatoriedade de utilizar informação sensível, como a localização ou ruído de ambiente. Esta abordagem endereça preocupações de privacidade e limita inconvenientes que o uso de sensores acarreta, como falhas na agregação de dados ou o gasto excessivo de bateria, ao não utilizar informação proveniente dos sensores do telemóvel.

Palavras Chave

mHealth; Notificações; Aprendizagem Automática; Personalização; Aprendizagem por Reforço; Receptividade.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

NCDs	Noncommunicable diseases
WHO	World Health Organization
USD	United States dollars
EMA	Ecological Momentary Assessment
ML	Machine Learning
SL	Supervised Learning
UL	Unsupervised Learning
SSL	Semi-Supervised Learning
AL	Active Learning
RL	Reinforcement Learning
LSTM	Long Short-term Memory
NN	Neural Network
G_r	Goal Achievement Rate
N_v	Notification Volume
DQN	Deep Q-Learning
UCB	Upper Confidence Bound
TQ	Tabular Q-learning

1

Introduction

Contents

1.1 Motivation

Noncommunicable diseases (NCDs), also known as chronic, such as cancer, diabetes, stroke, and other chronic respiratory or cardiovascular diseases, are the leading causes of death and disability worldwide. Nowadays, these represent more than 70% of all deaths and create devastating health consequences. This epidemic threatens to overwhelm health systems across the world, making it essential for governments to prioritize health promotion and disease management [2].

With the increase of life expectancy and consequent population aging, more and more people are expected to live with multimorbidities and NCDs [3]. Furthermore, with the Covid-19 pandemic, new challenges arise as countries struggle to maintain their NCDs diagnosis and treatment services undisrupted while hospitals get overcrowded and their staff overloaded. Additionally, patients fear that by going to healthcare facilities to receive their treatments, they could be at higher risk due to the possibilities of getting Covid than by just not receiving proper treatment for their chronic diseases. Many have now adopted telemedicine as an alternative, allowing people to still be followed and receive at least a minimal amount of support and supervision [4,5]. Nonetheless, self-management is now more vital than ever and many studies have already shown promise for its application in helping manage these chronic conditions [6–13].

1.1.1 Mobile Health Applications

Mobile health, or mHealth, was first defined by the World Health Organization (WHO) as "medical and public health practice supported by mobile devices" such as phones, wearables, patient monitoring devices, personal digital assistants, or other wireless devices [14] and is a great vehicle for the support of self-management in NCDs [15–18].

In Europe, the mobile market is one of the most advanced, recording in 2020 an average mobile cellular subscription rate of 122.7 per 100 inhabitants, well above the world average of 105 [19]. This growing penetration of the Internet and phones is significantly contributing to the growth rate of the mHealth market. Globally, the mobile medical apps market is projected to reach 156.82 billion United States dollars (USD) by 2026, from a value of USD 31.14 billion in 2020. Among the various categories, chronic care management is currently its second largest segment [20].

Mobile phones, now embedded into people's routines, are deemed a pervasive, ubiquitous technology, meaning they become almost invisible due to how natural their presence is to the user. By combining multiple sensors, which allow the capture of contextual, environmental, and behavioral data, with advanced Internet, storage, and processing capabilities, phones enable user profile customization and continual passive sensing (that does not require the users' direct interaction with any artifact or person). In this way, while disrupting as little as possible, smartphones make an appropriate hub for

monitoring systems. Data such as communication or speech characteristics, location, and activity can be objectively and frequently collected and processed to enable timely, personalized interventions that do not interfere with the user's everyday life [6, 21].

Complementary to passive sensing, smartphones can be used to gather Ecological Momentary Assessment (EMA) data through the repeated sampling of an individual's behaviors in their natural environment and in real-time, so as to be less subject to recall bias. EMA requires the collection of self-reported data through a variety of change-sensitive questionnaires. Many studies have actively exploited mHealth combined with EMA by providing questionnaires to simplify treatment accessibility, improve user retention, and qualitatively monitor symptoms and treatment progress [22–24]. This method does entail a non-negligible time commitment by the participants since it relies on daily, or even more frequent, user inputs.

Some key factors still restrict the adoption of mHealth, for instance, the lack of standards and regulations, privacy concerns, or the limited guidance and acceptance from traditional healthcare providers. The decisive factor, however, is always the necessity for widespread user adoption and continued engagement [10, 12, 16, 20, 25–27].

This thesis focuses on developing an intelligent notification system that intends to increase continued engagement by helping applications communicate with users only when they are receptive, instead of bothering them on inconvenient occasions.

1.2 Problem Statement

Phone notifications are widely employed to achieve user engagement, having been proven to significantly increase verified compliance when compared with identical trials that did not employ this technique [28–30]. However, the risk of intrusiveness into daily life is constantly imminent. Furthermore, consumers are known to highly dislike excessive or inopportune notifications, primarily when originated by machines [31, 32]. As previously stated, some of the most significant barriers to mHealth applications originate in the user itself. For this reason, mHealth applications must be able to function and communicate correctly without inducing further disengagement or burdening the consumer.

Notification systems that learn how, when, and where people engage with their phones allow for personalized communications, enhancing user experience. Systems such as these have been vastly studied in recent years, resorting primarily to machine learning techniques due to their capability of discovering structural and temporal patterns in data [21, 31, 33–36]. Notwithstanding, many challenges remain, mainly in three categories: technological, methodological, and privacy-related issues [29,31,37].

In this thesis, a notification system that learns a user's routine and preferences through reinforcement learning is designed and developed. This work addresses the challenges listed by not requiring any

passive or active sensing, hence dismissing sensor related issues, such as lack of precision, noisy data, or even battery drainage. Additionally, this approach diminishes privacy concerns since no sensitive data is collected and no storage or processing in external servers is required, maintaining any personal information in the users' device.

1.3 Thesis Outline

This thesis is organized as follows:

- Chapter 1 provides a brief presentation of this work's motivation and primary goals;
- In Chapter 2, the problem is framed by explaining some basic concepts and giving an overview of the current state of the art;
- Chapter 3 describes the practical approach that this work intends to develop and how it should be tested;
- In Chapter 4, the experiments to be executed are further detailed;
- Chapter 5 offers a presentation and discussion of the obtained results;
- In Chapter 6, a summary of this thesis' conclusions and an exposition of possible future work are presented;
- Lastly, in Appendix A, some supporting tables and graphs are shown.

2

Background

Contents

The purpose of this thesis is to develop a mechanism that is able to identify opportune moments for mobile device-based notifications. This goal entails challenging cross-disciplinary subjects such as information technology, medicine, and psychology, which plays a big part in understanding human behavior and preferences. Consequently, in this chapter, a review of the related work is presented in 3 key areas: the characterization of users' interaction with notifications, the development of smart notification systems, and the application of machine learning in the selected field of focus, mHealth.

2.1 Notifications and Opportune Moments

Push notifications emerged to contrast with the traditional pull-based information retrieval and delivery paradigm, where the user would have to be the one to initiate a request. Throughout an average day, users can now get over 50 notifications on their phones [31, 38, 39]. For that reason, feeling overwhelmed or even experiencing growing negative sentiments towards individual notifications or apps is expected [16, 31, 33, 40].

Grounded in previous studies, receptivity is defined as the degree to which a user considers that a notification is received at an opportune time. A high value implies that a user reads and interacts with the notification almost immediately. A low one indicates that the user feels interrupted and may ignore or discard it. Here, a time is considered to be appropriate or opportune when the user sees the notification and has sufficient time, energy, and motivation to address it and complete the associated required task.

Currently, there is yet no systematic way to infer user availability and receptivity. Identifying ideal moments for interaction with a permanently high level of accuracy is a complex problem due to its dependency on many aspects of a user's context [31, 33, 34, 41], such as location, movement, time, psychological state, personality traits, environment, hardware status, general device usage, or even simple demographics as gender or age. Nevertheless, many researchers have studied this matter. As a result of that work, a compilation of some common beliefs and assumptions regarding notifications and how people experience and react to them is presented.

A user's receptivity is highly influenced by how interesting, entertaining, relevant, and actionable the presented content is for him [42]. Additionally, researchers have proved that people feel less disrupted by notifications when bored, idling, or executing activities that demand a lower concentration level [34, 41]. It was also determined that receptivity should be treated as a preference that might differ from actual behavior, especially when self-defined or communicated by the user [31, 33, 43]. For example, users tend to estimate low receptivity levels for working hours because they do not want to be interrupted. However, if that preference was disregarded and notifications were still sent, they would react quickly and, if the content were considered interesting enough, some interruptions would be tolerated or even deemed desirable [41, 44]. In any event, studies have found that initiating interactions at unsuitable moments

might adversely impact ongoing tasks and psychological states. As a source of interruption [31], these interactions can negatively affect task completion time [45, 46], error rate [45, 47], and the affective state of the user [45].

Researchers commonly agree that notifications are most disruptive when the user is finishing or in the middle of a task [35, 41, 44, 48]. For this reason, when determining notifications' timings, applications should consider the type, completion level, and complexity of the user's ongoing task [41]. This is especially relevant since, on average, authors estimate that if the timing is opportune or the content relevant, users handle notifications in the first 10 minutes after initially seeing them. After that, it is unlikely that a notification is ever addressed [31, 38].

Studies have also indicated that people tend to handle notifications the fastest when close relatives send them. At the same time, messages from subordinates and systems (where the sender is not a person) are considered the most disruptive [31]. As a result, when developing any mHealth application, it should always be accepted that software cannot replace human contact and, as such, notifications are frequently perceived as disruptive.

Nearly all current applications employ a basic interaction model that assumes the user is available and willing to engage with the device at any time or at specific predefined time schedules,which the developer usually defines. However, it can be argued that the pervasive nature of mobile phones and their inherent constant accessibility does not necessarily translate into receptivity. In this way, the potentially disruptive impact of these applications can, and should, be compensated with the customization of notifications' characteristics (such as presentation or alert type) [41], content [16, 31], and an intelligent approach to deliver them [49]. Therefore, systems that attempt to handle such notifications intelligently are increasingly relevant.

2.2 Notification Systems

Intelligent notification systems, or interruptibility management, has attracted researchers well before the arrival of mobile devices. Initially, this subject focused on notifications received through desktops, which have very particular characteristics due to their typically static nature.

Mobile devices introduced a whole new dimension of trials as a result of the ease with which their surrounding environment and context can change. Additionally, a much higher number of notifications is received on these devices than on desktops [39]. Thus, interruptibility management for desktops is in many ways a less complex and dynamic matter and, for that reason, this review only aims to provide an overview of intelligent notification systems proposed mainly for mobile devices. Although many studies have been published on this kind of system, most of the existing prior work can be divided into two groups [36]:

1. **Detect transitions between activities**

   Based on the previously proved conclusion that users feel less bothered if notifications arrive once their former activity is complete and before starting another one, this group of studies aims to show notifications when a user is transitioning between activities.

   In [35], Ho and Intille propose that transitions between activities might suggest self-interruption moments when users would feel less disrupted if notifications were received. To detect these opportunities, they resort to wireless accelerometers that allow the detection of postural and ambulatory activity transitions. Two sensors were attached to the user, one to the outside of the right ankle and the other to the outside of the left thigh, just above the knee. Through the application of supervised learning, they were able to capture 4 types of transitions: sitting to standing, standing to sitting, sitting to walking, and walking to sitting. Their work demonstrated that interruptions delivered at any one of these switches tend to be judged by the user more positively. Nonetheless, they recommend that additional contextual cues be considered to help avoid poor interruption times entirely.

   In a different approach, Fischer et al. [48] proposed that the endings of episodes of mobile interaction represent natural breakpoints between tasks and consequently more opportune moments. Compared to notifications received at random times, the suggested concept was proven to obtain quicker response times. Yet, by analyzing the in-situ self-reports submitted by the user whenever a notification was received, the authors concluded that there was no improvement in the subjective experience between both conditions.

   Likewise, Okoshi et al. initially developed Attelia I [50] to defer notifications and deliver them only when appropriate breakpoints occurred. The researchers considered that these moments would be detectable in real-time by analyzing only smartphone user interaction events. Through the application of machine learning classifiers, models were created to test this hypothesis and tested in an in-the-wild user study formed by 30 participants and with a duration of 16 days. Although their results proved that users' cognitive load decreased when compared to randomly-timed notifications, they argued that many other, if not more, opportune times occur while the user is not actively manipulating the device.

   In consequence, the authors later developed Attelia II [51], where they incorporated their own suggestions into the system. This study used the above detailed interactions and added sensed data from multiple mobile and wearable devices to train the classifiers, thus including periods in which the devices were being carried but not actively interacted with. Taking into consideration the described data, transitions between activities were detected and used to send notifications. As a result of an in-the-wild evaluation in users' multi-device environments (smartphones and smartwatches) with 41 participants throughout a month, a 71.8% reduction of users' perception

of workload was achieved, thus reaching a more significant reduction than with Attelia I, which generated only a 33% decrease.

It is relevant to note that Okoshi el at. measured the results using self-reported nightly NASA-TLX [52] surveys. These apply the NASA Task Load Index, a widely used subjective assessment tool that rates perceived workload. The fact that they were only filled nightly imposes an evident limitation regarding the temporal distance between the actual notifications received and the filling of the survey. Furthermore, answers could be highly influenced by the users' experience throughout the day, which could lead to the submission of inaccurate evaluations of their receptiveness to individual notifications.

Unlike desktop interruptibility management mechanisms, which are highly focused on task transition phases, in mobile equipment, many of the user's tasks are not related to interactions with the device itself. It is, therefore, more complicated and sometimes even impossible to efficiently determine when these transitions occur [36, 49].

2. **Infer receptivity from the current context**

   Today, interruptibility studies emphasize the exploitation of features that are readily available to them through the wide variety of existing mobile sensors. As is visible in more detail in the next section of this chapter, numerous studies have already demonstrated the potential of this type of sensing to infer users' physical and psychological state, as well as their environment.

   Pejovic et al. developed InterruptMe [33] as an interruption management library for Android mobile devices. A mixed-method of automated smartphone sensing allowed the collection of contextual information and experience sampling by asking users about their interruptibility at different moments. With this information, classifiers were then built to identify opportune moments for interruptions. Over a two-week study, they collected users' in-the-wild context information such as location, physical activity, emotional state, and level of engagement with the ongoing task. Their results showed that the intended moments could be predicted with an average accuracy of about 60% when applying the created models.

   Contrasting with the first group of studies, Pejovic et al. did not find a significant correlation between interruptibility and activity change. Furthermore, the authors tested their classifiers with different combinations of the collected contextual data, employing different permutations of time(T), location(L) and activity(A), (T, L, A, T&A, T&L, A&L). They concluded that, when using the created models, the additional precision enabled by energy-expensive sensors (such as GPS) did not appear to be meaningful.

   When deploying the InterruptMe library with the developed classifiers instantiated within it and allowing personalization for each user throughout the application lifetime, they obtained a higher

overall sentiment and a faster response to notifications. Moreover, this last trial highlights interruption frequency as a critical factor in user sentiment towards notifications.

Mehrotra et al. [31] used a similar technical approach but focused on different indicators of availability. Here, researchers took into consideration a dataset which collected both content and context information, such as the users' current activity, the type of information delivered, and, when applicable, the social relationship between user and sender. Their experiences proved that, when their classifiers were deployed in an online fashion, more than 60% precision could be achieved after only nine days, outperforming subjective interruptibility rules defined by the users themselves. However, there were some limitations regarding how notifications were considered successful or ignored. If these required no further action or users could have attended to them on other devices, they would be misclassified as ignored, hence, training the model with incorrect data. On top of that, significant privacy issues were raised due to the necessity for personal information when establishing the users' personal circle.

In [34], Dingler and Pielot argued that boredom levels could be estimated through a combination of context information and mobile phone interactions. The central claim was that when bored, people would be more open to receiving notifications. A machine learning model was built to attest this theory, combining features as the recency of phone communication, the intensity of recent (last 5 minutes) and general phone usage, the time of day, light sensor values, and simple demographics such as age or gender. When applied, the model attained an accuracy of up to 82.9%. Additionally, the authors also demonstrated that users are more likely to engage with suggested content when bored.

In order to obtain their conclusions, the authors employed two different methods. First, a field study where mobile phone usage data and the ground truth about the participants' level of boredom was collected. With that data, classification models were trained and tested. Second, an online validation pilot study was performed, where the obtained models were applied, and specific entertainment content was provided when moments of boredom were detected. Seeing as many of the existing studies in interruptibility management are conducted with small population samples, in short periods, and typically performed in an offline fashion [49], the fact that this work included some form of in-the-wild online application of their models is already an improvement. Nonetheless, further work needs to be performed to provide more substantial statistical proof that the reported effects could be observed in different settings and populations.

2.3 Machine Learning in mHealth

Propelled by the development of learning algorithms, the constant expansion of available rich data, and the decreasing cost of computation power, Machine Learning (ML) is currently one of the fastest growing technical fields [53]. Advances in Cloud computing that support accessible on-demand data storage and computing power services further encourage its use in mobile applications.

There is no universally accepted definition; however, ML can be generally recognized as "an algorithmic framework that can provide insights into data while facilitating inference and providing a tentative setting to determine functional relationships" [30]. ML techniques can be divided into five types: Supervised Learning (SL), Unsupervised Learning (UL), Semi-Supervised Learning (SSL), Active Learning (AL) and RL [54].

For the design and implementation of intelligent interactions with mobile users, ML is the subject of numerous research works. The amount increases yearly, showing particular benefits in the field of digital health. SL is the most common method for this kind of smart interaction, presumably due to the tendency of seeing the problem at hand as the need to classify users, their preferences, or even labelling moments in time as opportune or inconvenient [54].

mHealth applications that utilize models derived from a combination of sensor data and self-reports can infer a person's physiologic [13, 23], psychological [13, 15, 23], and environmental state [23]. The research community continually develops and tests such apps in an attempt to address NCDs as diabetes [21], asthma [55] or cardiovascular diseases [13] and other health concerns like obesity [56], smoking cessation [57], stress management [58, 59], or depression treatment [23]. A deeper overview of these studies is presented in Table 2.1.

As a result of reviews of previous work [27, 32, 60, 61], authors highlight the need to create more intelligent, personalized, and efficient patient-centric mHealth models. For this purpose, ML is again a great approach due to its personalization capabilities [21, 62]. As of now, most tailoring approaches are applied at the group level instead of bringing it down to the individual level [63]. Furthermore, it is relevant to note that although personalization is recommended and considered beneficial, such customization makes applications more vulnerable to losing credibility whenever a miscalculation occurs [64].

Albeit its known capacity for swift adaptation to dynamic, unpredictable, and complex environments, RL is more complicated to apply than supervised or unsupervised learning, leaving it with few and modest implementation attempts in the mobile applications field [54]. Nonetheless, its ability to learn in real-time while attempting to maximize user satisfaction and without requiring the existence of a training dataset makes it a desirable study subject. It is recognized that a reinforcement learning model is composed of a set of environment states S, a set of possible actions A, and a set of scalar reinforcement signals, generally called rewards (R). In this method, an agent learns through interactions with the surrounding environment, resorting to a framework that defines them in terms of states, actions, and

rewards. It allows the definition of short or long-term goals that the agent achieves through a trial and error process.

In order to explore the potential of employing RL, in this work, the developed notification system is formulated as a RL problem, further detailing the above introduced components in the following chapters.

Paper	Project Name	Infers user state	ML technique	Disease
[23]	Mobilyze!	physiologic, psychological environmental	SL	Depression
[13]	HEARTEN	physiologic, psychological	SL	Heart Failure
[55]	-	physiologic	SL	Asthma
[57]	mPuff	physiologic	SL	-
[59]	-	psychological	SL	Stress
[21]	DIAMANTE	physiologic, psychological	RL	Diabetes
[56]	-	physiologic	SL	Obesity

Table 2.1: mHealth Related Papers.

2.4 Related Work

This section describes in deeper detail a set of papers with which this work shares a common purpose or technique and that spotlight the same areas of study already detailed throughout this chapter.

2.4.1 SELFPASS

In [43], M.-M. Theilig et al. had the primary goal of positively reinforcing self-assessment at moments of increased motivation. To achieve this, they combined environmental data with activity recognition to detect when users' receptivity was at its highest. Then, through SELFPASS, a therapeutic solution that supports self-management for patients with depression, they sent periodic self-reported questionnaires. Considering many of the same findings described earlier, they focused on detecting phases of idling or activities that demand a lower concentration level, predicting when a new activity starts and projecting an associated receptivity value.

In this paper, the approach to activity recognition was centered on the user's weekly schedule, modeled by sensor data. The authors choose this structure due to the belief that most people's activity patterns follow a week-based rhythm, exhibiting intra-day patterns for activities such as sleeping, getting up, eating, or coming home from work.

From the available data, sleep schedules, heart rate, flights climbed, distance walked or ran, and the number of steps were the ones taken into consideration. The number of activities recognized with

this data was limited to ten, all mutually exclusive and collectively exhaustive. A Long Short-term Memory (LSTM) model, a supervised learning technique that relies on recurrent neural networks trained by backpropagation, was selected. This model was utilized for both activity classification and receptivity projections. It used two main inputs: the sequence of activities already executed and detected and an auxiliary input vector, considered to provide helpful hints regarding the activities. The latter included data such as time of day, receptivity, and other integrated sensor information. The goal was that, from these vectors, the application of LSTM would allow the system to project a user's following day activities and respective receptivity values.

In an initial experience, the required data was collected and used to train and test the model. Here, data samples from the previous two activities as well as the current one were utilized to predict the upcoming activity. On average, the authors were able to correctly predict about 39% of the day, achieving 58% in their best cases. Self-reported receptivity values were used as input, but no receptivity projections were presented in the paper. However, they argued that since receptivity values varied immensely for each individual, linking their prediction to each user's preferences would be the more promising approach. They also confirmed that even for a limited data set, people's daily activities show reproducible patterns.

In an attempt to provide a more enjoyable experience, when developing SELFPASS, they allowed users to fine-tune activity–receptivity pairs. Individual values of receptivity were customizable for each detected activity, as was the threshold which defined what level of receptivity implied an opportune timing for the user. Users could also set their individual standard of being active, for example, a personalized heart rate threshold for when they exercise, enhancing the system's performance even further.

The user's reaction to notifications was registered but not utilized in any way. Nevertheless, in the app, the user could reinforce times that would have been associated with good receptivity. Unfortunately, this feature was only available from the ex-post point of view for moments already passed.

Although M.-M. Theilig et al. developed SELFPASS according to the results of the first experiment; no further experiments were executed to test the implementation of the system in its entirety, leaving many doubts regarding the projection of receptivity values and any improvements that the added customization may have added.

2.4.2 Nurture

With a similar objective but applying a different technical approach, Ho et al. [36] proposed an RL-based personalization method named Nurture. Through the analysis of user context, it detected appropriate times to send notifications, independently of the content included. To do so, the authors decided to consider the day (weekday or weekend), time (morning, afternoon, or evening), location, motion activity (stationary, walking, running, or driving), and time since the last notification (within or beyond 1 hour).

The authors wanted to demonstrate that the implementation of RL would provide better results than the typically used supervised learning methods. This belief was mainly based on the fact that this ML technology enables the learning of user preferences simply by interacting with them over time, without requiring any prior knowledge. With this goal in mind, they implemented two well-established algorithms: one as a contextual bandit and another representing a full reinforcement learning problem. The main difference between these approaches is that a contextual bandit does not learn transitions between states.

Two experiments were designed, and each was carried out over the course of 12 weeks to evaluate their implementation: a synthetic and an online interactive crowdsourcing-based simulation. For the latter, Nurture interactively and in real-time sent notifications in the form of a survey to users throughout India and the United States. It should be noted that in this simulation, only one of the algorithms was tested. For the former synthetic trial, the authors developed a simulator that reproduced a user's context and daily routine, hence providing the required information for the agent to decide if a notification should be sent or not. Surveys describing the scenario of a specific user situation were shared on Amazon Mechanical Turk7 and asked users to decide if, in that context, they would answer a notification. Based on the collected responses, the simulator was then able to report if a user would answer the received notification or not.

Compared to a baseline calculated through the application of Support Vector Machines, the application of RL provided better results. Overall, this research demonstrated that this approach substantially improves the notification response rate while balancing the number of notifications.

2.5 Contributions

The methodologies presented throughout this chapter support this thesis's intention of applying ML in interruptibility management. However, these works tend not to discuss in detail the formulations of their problem and always resort to the use of sensors and possibly sensitive information to implement their methodology.

This work's contributions entail the development of a RL-based intelligent notification system that employs the available information in the phone's system itself, resorting to none of the device's sensors. The primary source of information consists of notification interaction behavior, meaning the history of how the user has reacted previously to similar notifications, hence, not using any private user information. Furthermore, in this thesis, the question of whether opportune timings show a daily or weekly pattern is also going to be addressed.

3

Methodology

Contents

This chapter presents our approach to investigating the efficiency of an intelligent notification system, built as an RL-based agent that receives as primary input notification interaction behavior, optimizing phone resources and users' privacy by avoiding the inclusion and usage of sensitive user data as part of the state definition. It also illustrates the design and implementation of the simulator utilized to test this methodology and obtain the results discussed in Chapter 5.

3.1 Proposed Approach

This thesis envisions the search for opportune notification timings as a reinforcement learning problem. Through the application of RL, our system aims to learn user preferences, routines, and habits merely from notification interaction data.

Our work is centered around mHealth and improving continued user engagement. Hence, the main aim is to discover one moment throughout the day when the users are available and willing to answer a notification that leads to the required action. This action could be any self-management task required by any mHealth application. When this goal is achieved, no more notifications should be sent. For that reason, the RL agent considers an answered notification as a terminal state, meaning that the episode, in our context, a day, has ended and that the agent only starts working again when the next day begins. The only other terminal state occurs at the end of the day (24h). A more detailed explanation of what an episodic task, such as ours, implies is provided in Section 3.2.1.

In contrast with previous works, here it is proposed that RL is better applied to our problem than SL techniques for reasons such as:

- RL is able to learn and provide efficient results purely from online learning, without any prior knowledge or training required;
- In scenarios as dynamic as human context and preferences, it is essential that a system such as this can adapt to new states and routines. RL often explores different actions, mainly when unseen observations or different reactions are obtained;
- In RL problems, agents can be designed with the intention of ensuring the maximization of future rewards over immediate ones.

Figure 3.1 describes the architecture of the proposed system, which depicts the learning flow of our agent. In this thesis, the environment is implemented by a simulator that generates user responses following a probabilistic model. The development of the simulator is explained in further detail in Section 3.5.

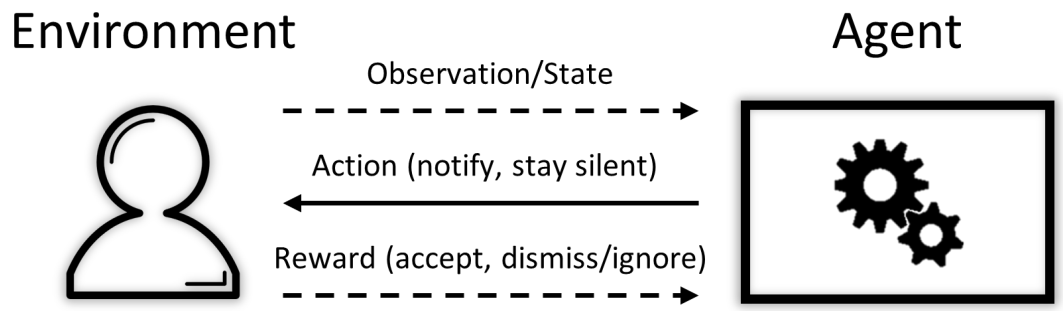


Figure 3.1: Interaction between the user, in their natural environment, and the learning unit.

The main objective is that, through observing the user and the environment's state, the agent should decide whether to send a notification or stay silent. Then, if a notification is sent, the agent observes the user's reaction and continually learns from it. In our case, accepted notifications denote positive signals while dismissed or ignored ones are seen as negative reinforcement signals, penalizing the agent for the action taken. The agent's behavior changes accordingly, always intending to increase the long-run sum of rewards (reinforcement signals). These components allow the creation of a sense of cause and effect, the existence of explicit goals, and help the agent deal with the environment's uncertainty and nondeterminism. A goal is expressed by the reward structure, which, as previously mentioned, is to get the user to answer one notification per day.

An essential concept in RL is the idea of value estimates, also named value functions. Whereas rewards reveal what good behavior is in a more immediate sense, values indicate what is better in the long run. The former represents an estimate of the sum of rewards an agent could accumulate from then on, indicating their long-term desirability by considering the state-action pairs that are expected to follow and their respective rewards. This notion is very relevant because although a state-action tuple might be associated with low rewards if usually followed by tuples that achieve high rewards, the former would have a high value estimation. Even though rewards and value estimations are distinct concepts, they are intertwined. Without rewards, no values could be calculated, and the sole aim of value estimation is to attain higher rewards. Notwithstanding, the agent usually chooses their actions based on values estimates since its primary goal is to maximize future rewards. Unfortunately, values are much harder to determine as, contrary to rewards that are given directly by the environment, these must be continuously estimated and re-estimated over an agent's lifetime.

The best combination of algorithm, rewards and state definitions must be found to discover the most efficient solution for this learning problem. For that reason, this work reviews, selects, and tests several combinations. Additionally, a comparison between the obtained results amongst themselves as well as to a benchmark obtained from previous related studies is executed and detailed.

3.2 RL Algorithms

In this section, some known conflicts and initial concepts of RL are explained in order to help us better define the task at hand and present the applied algorithms.

3.2.1 Episodic Tasks

Due to how our problem is conceptualized, there are clear moments when the agent–environment interaction breaks naturally into subsequences. In the RL field, these subsequences, called episodes, embody repeated interactions that end in a terminal state and are followed by a reset to a standard initial state. For us, an episode represents a day. Episodes can end differently, but the following one always starts independently of how the preceding ended. Tasks such as ours can be described as episodic tasks.

3.2.2 Exploration vs. Exploitation

One of the most relevant characteristics of RL that differentiates it from other types of learning is that it learns without needing previous knowledge on what is a right or wrong action. By evaluating signals received after taking specific actions, the agent can start to understand if an action was good or bad but not if it was the best under those circumstances. To do so requires active exploration of what good behavior represents for the environment.

If you save estimates or averages of the expected reward associated with each action, you can always select one whose estimated value is better. These are named greedy actions. When selected, the agent is said to be exploiting current knowledge. Contrarily, if a nongreedy action is taken, then the agent is said to be exploring.

Exploiting might easily maximize the expected reward, but exploring may lead to even higher reward values if a less experimented but better state-action pair is available. Furthermore, exploring enables the improvement of estimates associated with nongreedy actions, reducing the agents' uncertainty level. Especially in nonstationary tasks like ours, where the preferred actions may change over time, exploration is essential to ensure that a nongreedy action has not changed to become better than a greedy one. This tradeoff highly influences algorithms' efficiency and demands a deep analysis of the problem at hand before deciding how to attempt to balance it.

A simple manner of dealing with the conflict between exploration and exploitation is to always select greedy actions. However, in the long run, this method has been proved to perform poorly compared to others because it often gets stuck executing suboptimal actions [65]. Another alternative may be to behave mostly in a greedy way but occasionally, and with a small probability ($\varepsilon > 0$), select a random action from the entire action space disregarding all action value estimates. This selection rule is called

the ε-greedy method. Many others exist, such as softmax, Minimize Collective Unhappiness, Value-Difference Based Exploration, or even combinations of these amongst themselves [66, 67].

The best solution is not always the same but the one that better suits the problem at hand. However, the ε-greedy is considered an overall efficient method, working as well as other more sophisticated and challenging to implement techniques. Hence, and after some testing in order to ensure that it worked in this thesis context and implementations, ε-greedy was the chosen method when action selection rules were required.

3.2.3 On-policy vs. off-policy

The exploration versus exploitation conflict causes an additional yet to be mentioned dilemma. Agents seek to learn what optimal behavior is but need to pursue less optimal actions in order to explore the environment. How can an optimal policy be learned while simultaneously applying an exploratory one? Two options emerge.

On-policy methods are somewhat of a compromise where only one policy exists. Instead of striving for the optimal policy, it learns values for a near-optimal one that never entirely stops exploring. The second and more straightforward technique is off-policy methods, where two policies exist. A Target policy that the agent learns about, strives to improve, and is the optimal one, and a Behavior policy, which is the exploratory one used to generate behavior, as the name suggests. Hence, the target policy learns its estimates from data generated by the separate behavior policy. A benefit of this separation is that the former may be deterministic (for example, a greedy policy), while the latter can still continuously sample equally from all possible actions.

When off-policy methods are applied, the learner still explores, but it can concurrently learn a deterministic optimal policy that is not necessarily related to the actual followed policy. This method was the one implemented through the chosen algorithms.

3.2.4 Nonassociative vs. Associative Settings

In a simplified perspective, nonassociative settings can be applied and have been widely studied as they avoid much of the complexity that a full RL problem brings. These are often called bandit problems and signify that the agent learns to act only in a single state setting. Nevertheless, generally speaking, tasks have more than one situation, and for those, the goal is to learn a mapping, also called policy, from specific contexts to the respective best actions.

Tasks that entail both trial-and-error learning and associating actions taken with the situations in which they are most appropriate are named associative search tasks or contextual bandits. These appear as an intermediate between the bandit problems and the full RL problem.

In this work, since the desire is to implement various views of the RL problem, both nonassociative and associate settings were developed.

3.2.5 Bellman Equation

The Bellman equation is one of the central elements of various RL algorithms. It essentially decomposes the value function into 2 sections, the immediate reward and the discounted future state-action values yet to be received, allowing for a derivation of a relation between state values (V) and action values (Q), as shown in equation 3.1.

$$\begin{aligned} Q_\pi(s,a) &= \sum_{s'} P_a(s,s')(R_a(s,s') + \gamma V_\pi(s')) \\ \text{with} \quad V_\pi(s) &= \sum_a \pi(s,a) Q_\pi(s,a) \end{aligned} \tag{3.1}$$

Where:

a - Action; s - State;

V - State-value function,defined as the expected empirical reward for each state;

π - Policy, representing a mapping of specific states and their respective action values;

Q - Action-value function, defined as the expected empirical reward for each action;

P_a - Probability of taking action a from state s to new state s';

R_a - Reward from taking action a in state s and moving to state s';

γ - Discount factor, explained in the next section.

The Bellman equation simplifies the computation of the value function in such a way that rather than summing over multiple timesteps, an optimal solution for complex problems can be found by simply breaking it down into smaller, recursive, easier to solve subproblems. Furthermore, it allows the direct approximation of both an optimal action-value function, equation 3.2a, and an optimal state-value

function, equation 3.2b.

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'}(q_*(S_{t+1}, a')) \mid S_t = s, A_t = a\right] \\ \Leftrightarrow q_*(s,a) &= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'}(q_*(s', a'))\right] \end{aligned} \tag{3.2a}$$

$$\begin{aligned} v_*(s) &= \max_{a \in A(s)} q_{\pi_*}(s,a) \\ v_*(s) &= \max_{a} \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')] \end{aligned} \tag{3.2b}$$

Where:

A, a - Action; S, s - State; R - Reward;

v_* - Optimal state-value function;

π - Policy, representing a mapping of specific states and their respective action values;

q_* - Optimal action-value function;

γ - Discount factor, explained in the next section.

3.2.6 Algorithms Included

In order to perform experiments that are as varied as possible, several well-known, tested, and consensually recommended algorithms were implemented. These encompass different views of RL, from a simple n-armed bandit problem, nonassociative, to an associative setting, and even what can be considered a representation of a full RL problem. Based on our research, three algorithms were selected to be applied:

- **UCB**

  UCB, detailed in Algorithm 3.1, emerges as a widely accepted bandit, as it considers the problem as only a single state. While ε-greedy methods choose arbitrarily, UCB chooses deterministically but still achieves exploration by subtly favoring the selection of actions that have the potential to be optimal and have been employed the less [65]. To do so, it applies the selection rule found in equation 3.3.

$$A_t \doteq \underset{a}{argmax} \left[Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}} \right] \tag{3.3}$$

where:

A, a - Action; t - Current timestep;

Q - Action-value function, defined as the expected empirical reward for each action;

c - Confidence level that controls the degree of exploration;

$N_t(a)$ - Number of times action a has been selected prior to time t.

In equation 3.3, the term inside the square root represents the uncertainty of the estimates of action values, making A_t an upper bound of the probable value of each action. When action a is chosen, $N(a)$ rises, diminishing the associated uncertainty. Additionally, as t increases but $N(a)$ does not, the uncertainty of action a increases as well. Using the $\ln$, UCB guarantees that these increases are unbounded but get smaller as time goes on.

UCB ensures that, given enough time, all actions are eventually performed. However, actions with lower estimates or already frequently selected are chosen with decreasing frequency. This guarantees that the agent explores the action space in order to properly learn what is the best action for the problem it is dealing with.

Through the application of this algorithm, the agent either strives to find the best action or attempts to track it as user preferences change over time, making it applicable to both stationary and nonstationary tasks. Although very efficient, this algorithm has been proven to have performance issues when utilized to address more complex nonstationary problems which possess large state spaces [65].

Algorithm 3.1: UCB

begin

First $numActions$ turns: initialize experimental means by pulling each arm once

for $t \leftarrow numActions + 1$ **to** $numTurns$ **do**

for $a \leftarrow 1$ **to** $numActions$ **do**

Calculate $A_t(a, t)$ - shown in Equation (3.3)

Pull arm a that maximizes $A_t(a, t)$

Update sum of rewards according to the received reward and number of selections of the arm pulled, hence updating the action-value function Q

It should be noted that since UCB is a single state algorithm, it learns what is better suited for that state only, which in this thesis' context would be very restrictive. Hence, this algorithm is applied in two manners that are expected to provide efficient results by leveraging the advantages of UCB but still considering that the user goes through different states throughout time.

UCB Day is the first approach, where a different UCB instance is applied to each hour of the day, allowing the agent to learn what action is better suited for each decision point, which in our case is hourly. UCB Week is the second, more personalized approach, where a separate UCB instance is applied to each hour of each day for each day of the week. Meaning that considering that the week has 7 days, and 24 instances of the algorithm are created for each day, UCB Week combines 168 instances, learning what is the most beneficial action for each decision point, according to the weekly routine of a user.

- **Tabular Q-learning (TQ)**

Tabular methods are settled on the core idea of RL in its most straightforward format, that both the state and action spaces are small enough for the estimates of action values and their mapping with specific contexts to be represented as arrays or tables. Nowadays, these methods are some of the most commonly utilized due to their simplicity and ease of computation [65].

Q-learning was initially defined by Watkins in [68, 69] and can be defined by the following expression:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a (Q(S_{t+1}, a)) - Q(S_t, A_t) \right] \tag{3.4}$$

where:

A, a - Action; S - State; R - Reward; t - Current timestep;

Q - Action-value function, defined as the expected empirical reward for each state-action pair;

α - Learning rate; γ - Discount factor.

The learning rate, α, determines when Q-values are updated, overriding older information. A factor of 0 implies the agent learns nothing, relying solely on whatever knowledge it had previously obtained, while 1 would mean that the agent considers only the most recent information.

The discount factor, γ, models the relevance of future rewards by causing them to lose their value over time so that more immediate ones are valued more highly. A value of 0 makes the agent consider only immediate rewards, while 1 would make the agent strive for long-term high rewards.

As seen above, the equation (3.4) maximizes the next state-action pair, meaning that the policy is

greedy. It does not take into account how probable each action actually is under the current policy. Below, in Algorithm 3.2, Q-learning is presented in a procedural form.

Algorithm 3.2: Tabular Q-learning

begin

With $\alpha \in (0, 1]$ and $\gamma \in (0, 1]$

Initialize Q-table

for $episode \leftarrow 1$ **to** $numDays$ **do**

Initialize state S

while S is not terminal **do**

Choose action A from S using policy derived from Q - for previously unseen S choose A randomly

Take action A, observe R,S'

Calculate $Q(S, A)$ - shown in Equation (3.4)

Update Q-table with new knowledge

$S \leftarrow S'$

It should be noted that although equation 3.4 assumes a greedy policy, many different policies can be applied when actually choosing the desired action. As previously stated, this choice should take into consideration the context and particularities of the problem in question. In our work, the ε-greedy policy, introduced in Section 3.2.2, showed promising results and was henceforth applied. It is proved that Q directly approximates the optimal action values independently of the policy being followed [65]. Nonetheless, the policy in place still determines which state-action pairs are updated and selected.

When working with larger state spaces, tabular methods such as this are not appropriate due to the associated memory requirements and the amount of time and data needed to fill these tables correctly. Especially since, in many problems, most of the encountered states can be new to the agent, rendering the already saved pairs fruitless.

Unlike similar approaches, this algorithm is an off-policy method, meaning that, although not true, it assumes a greedy policy is being used. That can be verified in equation 3.4, where it is visible that Q is updated using the value of the following state and the value of the greedy action a, instead of the value of the real action taken. Off-policy algorithms often present more significant variance values as well as slow convergence times when compared to on-policy ones. Even so, they tend to be more powerful and easier to apply to more varied contexts and problems.

- **DQN with Experience Replay**

Developed by Mnih et al. in [70], the Deep Q-learning agent, described in Algorithm 3.3, combines

the previously described Q-learning algorithm with a Neural Network (NN). This network is usually a deep convolutional NN due to its many layers and fully connected network (meaning that each neuron in one layer connects to all neurons in the following layer).

Here, the agent's brain is the NN instead of a table or array. It receives an observation and outputs the estimated values for each of the available actions. It is updated through the mean square error loss function, where the difference between the current predicted Q-values (Q_θ) and the true value (Q_{target}) is computed according to:

$$Q_{target}(t) = \begin{cases} r_t, \ for\, terminal\ \phi_{t+1} \\ r_t + \gamma \max\limits_{a'}(Q_\theta(\phi_{t+1}, a')), \ for\, non-terminal\ \phi_{t+1} \end{cases} \tag{3.5a}$$

$$Loss(\theta) = \sum \left(Q_{target}(t) - Q_\theta(\phi_t, a_t)\right)^2 \tag{3.5b}$$

where:

a - Action; ϕ - State; r - Reward; t - Current timestep; γ - Discount factor;

Q - Action-value function, defined as the expected empirical reward for each state-action pair;

θ - Network weights.

Algorithm 3.3: DQN with Experience Replay

begin
- Initialize replay memory D
- Initialize action-value function Q with random weights θ
- **for** $episode \leftarrow 1$ **to** $numDays$ **do**
  - Initialize sequence $s_1 = x_1$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  - **while** S is not terminal **do**
    - With probability ε, select a random action a_t; otherwise select $a_t = \max\limits_a Q^*(\phi(s_t), a; \theta)$
    - Take action a_t, observe the reward r_t and following state x_{t+1}
    - Set $s_{t+1} = s_t$,a_t,x_{t+1} and preprocessed $\phi_{t+1} = \phi(s_{t+1})$
    - Store transition (ϕ_t,a_t,r_t,ϕ_{t+1}) in D
    - Sample random minibatch from D
    - Set Q_{target} for the selected sample - shown in Equation (3.5a) - and update the NN - shown in Equation (3.5b)

While this type of NN allows for more flexibility, it sometimes comes at the cost of stability. For that reason, many extensions of this algorithm have already been designed and tested. One, in particular, is called Experience Replay.

Experience Replay [71, 72] is a technique where the agent memorizes the state, action, and effect of that same action in the environment for every timestep. After completing an episode, it then replays the experiences gathered in its memory by randomly selecting a batch of a particular size and training the network with it. This replay helps reduce instability produced by training on highly correlated sequential data and increases the learning speed of the network.

3.3 State Representation

Generally, applying RL in real-world situations can be very complex since agents must gain efficient representations of the environment, usually from high-dimensional sensory inputs, and use them to generalize their experience to new circumstances. In cases such as the one in this thesis, even if the agent had a complete and accurate model of the environment, it would typically be unable to use it due to memory and computation limitations.

Ideally, a notification system would have access to the current activity, emotional state, location, and other private information that we prefer not to utilize in order to minimize privacy issues. Here, the focus is on using accessible notification information such as the time or day of the week, the user's reaction to the notification, or the number of notifications already answered. Hence, the aim is to demonstrate that very efficient results can still be obtained from more simplistic representations of a user's state. Thus, for the DQN and Q-learning algorithms, the state representations presented in Table 3.1 were designed. For UCB, and as previously stated, since the algorithm considers the problem to have a single state, such representations are not required.

State Type	Structure	Range of Values	Number of Possible States
S1	[time in minutes, number of notifications sent, number of notifications answered, last user reaction, day]	1440 minutes, maximum of 24 notifications, maximum value corresponding to the notification goal, 0/1/3 depending on the outcome of the last action, day of the week represented by values from 0 to 6	725760 (when notification goal = 1)
S2	Array of length 25	For the first 24 positions, 0/1/3 depending on the outcome of the action chosen at each hour of the day (initially, all positions start as 0). Plus 1 additional position representing the day of the week with values from 0 to 6	$1.98 * 10^{12}$
S3	[time in minutes, number of notifications sent, number of notifications answered by the user, last user reaction]	1440 minutes, maximum of 24 notifications, maximum value corresponding to the notification goal, 0/1/3 depending on the outcome of the last action	103680 (when notification goal = 1)
S4	Array of length 24	For every position, 0/1/3 depending on the outcome of the action chosen at each hour of the day (initially, all positions start as 0)	$2.83 * 10^{11}$

Table 3.1: State Representations.

Table 3.1 shows that only elementary and easily obtainable information is considered, focusing primarily on knowing how far the agent is from its objective and how the user reacts to its actions. Furthermore, since the generic goal is to learn the most opportune timings, time and, in some cases, even the day of the observation are also tracked.

These representations were created in a way that would always allow the agent to know the number of notifications already answered, ensuring that, if desired in the future, they are suited for similar problems where the goal is not limited to one notification per day. For our problem, the system was configured to only dispatch one notification per hour in order not to bother the user with repeated and closely followed alerts. This intention led to the creation of arrays of only 24 elements, which could quickly be narrowed or extended to incorporate other time intervals.

With the purpose of recording users' reactions, or lack thereof, to the actions of the agent, 3 options were defined and associated with a value in the following manner:

- 0 $\rightarrow$ In the last timestep, a notification was sent and ignored or dismissed by the user;
- 1 $\rightarrow$ In the last timestep, a notification was sent and positively addressed by the user;
- 3 $\rightarrow$ In the last timestep, a notification was not sent.

3.4 Reward Definition

The concept of reward and its connection to goals and value functions is already detailed in previous sections. This section further details the relevance of adequate reward definition and displays the various reward structures applied in this thesis.

At each timestep, the received reward is a scalar – $R_t \in \mathbb{Z}$. The efficiency of any RL application is dependent on how accurately these numbers frame the goal that the agent should achieve and assess progress in reaching it. Hence, designing reward signals is one of the most critical tasks in implementing an RL technique. For instance, in the case of games with pre-defined scores, the definition of different rewards is straightforward, but some cases involve goals that are more challenging to translate into signals. Furthermore, it is not uncommon that, when rewards are poorly structured, agents find unexpected ways to get higher rewards, some of which might be undesirable.

Rewards must be structured in a way that undoubtedly indicates what should be accomplished by the agent, not how to achieve it. Subgoals should not be rewarded to ensure that the agent does not find a way to achieve them without actually ever attaining the real objective.

A balance must be found between not rewarding subgoals while also not incurring on the sparse rewards dilemma. Providing non-zero signals frequently enough is a concern that often arises since

state-action pairs that should trigger rewards may be few and far between, making the agent act aimlessly for lengthy extensions of time.

Defining a non-sparse signal that efficiently describes our goal and guides the agent to it was a trial-and-error process. Several options were initially tested; if the agent did not learn, learned slowly, or ended up behaving differently from what was intended, then that option was excluded. Below, in Tables 3.2 to 3.5, the final selection of rewards that were applied during our experiments can be found.

Event \User Reaction	User does not answer	User answers	No reaction is possible
A notification is not sent	-	-	-1
A notification is sent	-1	2	-
Episode ends without the goal being achieved (24h)	-2		

Table 3.2: Reward 1.

Event \User Reaction	User does not answer	User answers	No reaction is possible
A notification is not sent	-	-	-1
A notification is sent	-2	2	-
Episode ends without the goal being achieved (24h)	-3		

Table 3.3: Reward 3.

Event \User Reaction	User does not answer	User answers	No reaction is possible
A notification is not sent	-	-	0
A notification is sent	-2	2	-
Episode ends without the goal being achieved (24h)	-3		

Table 3.4: Reward 5.

Event \User Reaction	User does not answer	User answers	No reaction is possible
A notification is not sent	-	-	0
A notification is sent	-3	2	-
Episode ends without the goal being achieved (24h)	-3		

Table 3.5: Reward 6.

The general idea we wish to transmit to the agent with these structures is that the goal is to get the user to answer one notification without bothering them, meaning without sending notifications that go unanswered. In this context, not sending a notification can be seen as a subgoal that we do not wish to reward but may penalize in order to motivate the agent to find the most opportune time for the user to

answer as soon as possible in the day.

Once again, like in Section 3.3, a new element could easily be added to these structures to define the reward provided when the new goal, for instance, 3 answered notifications per day, is achieved. In our case, since this thesis strives only for one a day, the element where a notification is sent and the user answers represents that reward. Of course, if changes were to be made, it would be essential to review the remaining values to assure that no subgoals were being rewarded enough to prevent the agent from achieving its primary goal.

3.5 Environment Model - Simulator Development

In our experiments, users are simulated by combining underlying models which approximate human responses. The creation of this simulator allows us to repeatedly and systematically iterate over various algorithms.

Through the simulator, the daily routine of a user is mimicked. In the future, instead of resorting to this simulated routine, it would be ideal to deploy our models in devices where they could interact with real users.

Depending on the algorithm, previous user responses, or any other initial information, the agent decides whether to send a notification or remain silent. When it decides that a notification should be sent, the simulated user is the one who determines if it responds or not. Here, this work does not differentiate between ignoring the message (performing no action to skip it) or explicitly dismissing it, considering both as “No Answer” received. This approach is followed because we do not wish to understand why a moment is less opportune but simply that it is. In this way, the users’ answers or lack of it are registered, and their motivations disregarded. Furthermore, the user’s answer is considered to be either immediate or non-existent.

To emulate the behavior of a regular user, both a behavior model and a response model are here designed and combined.

3.5.1 Behaviour Model

The first one reflects a user’s routine, for example, the activities performed, their duration, and the user’s location throughout them. This model mirrors the ExtraSensory dataset [1], which aggregates the daily traces of 60 participants.

Thousands of measurements from users’ smartphones and smartwatches were collected, along with self-reported labels, to form this dataset. Since the authors collected this data in the wild, its reliability is not perfect; there were missing and even wrongly reported labels. After processed and cleaned, it considers 51 possible tags, including 15 locations, 8 primary activities, and 28 secondary ones.

Primary activities describe movement or posture and are mutually exclusive, while secondary activities represent a more specific context. For the latter, such as for locations, the user could apply several tags to a single instance in time.

In this simulator, the users' state is represented accordingly, as the combination of one primary activity and a set of up to 43 possible secondary tags, composed by secondary activities and locations.

From the available data, three user traces were chosen to be used throughout our experiments. These were selected according to two main concerns: providing lifestyles as distinct as possible while ensuring the availability of enough data to represent a week in these users' lives. Relevant statistics regarding their weekly routines can be found in Appendix A.2.

3.5.2 Response Model

The second model created was the response model, which simulates how a user would respond to a notification in any given context. This model is responsible for the observations that our agent receives throughout the simulation. Based on our investigation, a set of behaviors that researchers consensually agree users will or at least tend to show were considered when implementing this model [16, 31–33, 38, 40, 43]. For instance:

- When the behavior model presents labels such as sleeping, which ensure an inability to answer, the simulator does not respond to notifications;
- For labels that represent tasks such as driving or being in a meeting, for which usually a low probability of answering is associated, the simulator tends not to respond;
- A randomness level is always associated with every decision the simulator makes, except when the user is sleeping. This level intends to express the same randomness a human would show in their daily life;
- A component (β), defined as the exponential decay visible in equation 3.6, is used to convey the growing irritation and consequent diminishing desire to use the app that most users would experience as the number of daily notifications rises.

$$\beta(n_t) = P(Answer \mid n_t) = e^{-\lambda n_t} \tag{3.6}$$

Here, n_t represents the number of messages already sent during the current day. λ equals 0.3, chosen to guarantee reasonable values are obtained. The obtained graph for the β component can be seen in Figure 3.2.

For the maximum number of notifications the system could send, $n_t = 24$, one per hour of the day, this component equals approximately 0.0008, producing an almost null probability of answering.

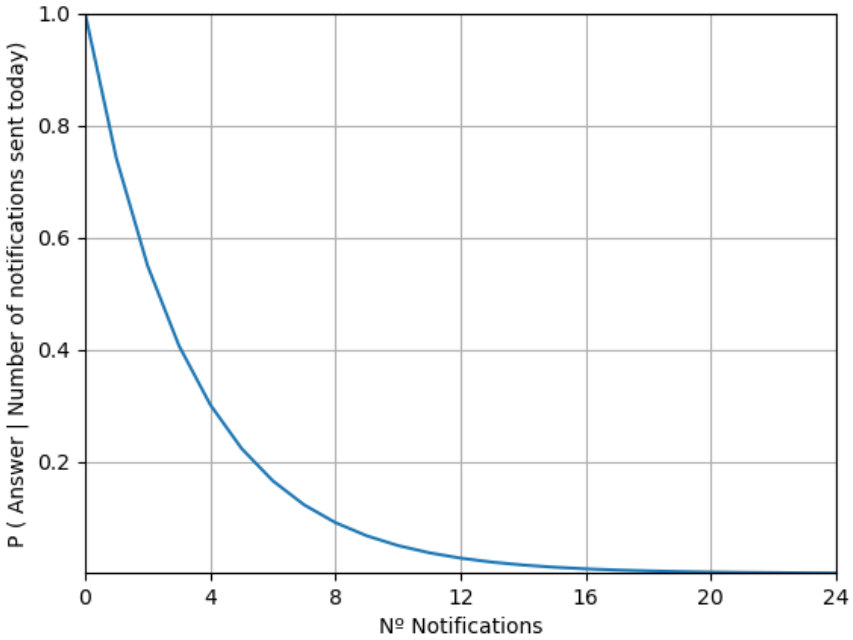


Figure 3.2: β Component in the simulated user's response model.

- Each user has a predefined prior probability of answering $P(A)$ and not answering notifications $P(\overline{A})$. This value represents a person's predisposition to be on their phone and regularly use an application such as the ones focused on in this thesis. For this reason, these are fixed values selected depending only on how compliant or not our simulated user ought to be.

Assuming statistical independence between labels and following the Naive Bayes probability model (equation 3.7), the probability of the user answering or not, given their current context, is calculated. A set of L_t labels, provided by the behavior model (as explained in Section 3.5.1) represents this context. Thus, for every instant in time, there are i labels that describe the moment (L) and $(L_t - i)$ that were not chosen and indicate activities/movements the user is not currently doing ($\overline{L}$). Considering our two possible classes (C), $Answer(A)$ and $NoAnswer(\overline{A})$, the model is formulated as follows:

$$\begin{aligned} P(C \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) &\propto P(C, L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \\ &\propto P(C)P(L_0 \mid C)...P(L_i \mid C)P(\overline{L_{i+1}} \mid C)...P(\overline{L_{l_t}} \mid C) \\ &\propto P(C) \prod_{j=0}^{i} [P(L_j \mid C)] \prod_{k=i+1}^{l_t} \left[P(\overline{L_k} \mid C)\right] \end{aligned} \tag{3.7}$$

The values of $P(L \mid C)$ and $P(\overline{L} \mid C)$ are unknown. To calculate these, the Bayes' theorem, shown in equation 3.8, is applied.

$$P(L \mid C) = \frac{P(C \mid L)P(L)}{P(C)} \tag{3.8}$$

Additionally, knowing the conditional probability formula (equation 3.9) and equation 3.10, it is possible to calculate $P(\overline{L} \mid C)$ using only $P(L \mid C)$.

$$P(C \mid L) = \frac{P(C, L)}{P(L)} \tag{3.9}$$

$$\begin{aligned} & P(C, L) + P(C, \overline{L}) = P(C) \\ \Leftrightarrow & P(L \mid C)P(C) + P(\overline{L} \mid C)P(C) = P(C) \\ \Leftrightarrow & \left(P(L \mid C) + P(\overline{L} \mid C)\right) P(C) = P(C) \\ \Leftrightarrow & P(L \mid C) + P(\overline{L} \mid C) = 1 \\ \Leftrightarrow & P(\overline{L} \mid C) = 1 - P(L \mid C) \end{aligned} \tag{3.10}$$

Leaving now only the values of $P(C \mid L)$ and $P(L)$ as unknown. Hence, these were transformed into either obtainable components from the behavior's model dataset or reasonably estimated.

3.5.2.A Estimation of Conditional Probability Values

For each label provided by the behaviour model, reasonable values were defined for the probability of answering given that label ($P(A \mid L)$) and not answering given that same label ($P(\overline{A} \mid L)$). These values are detailed in Appendix A.

3.5.2.B Calculation of the Probability of each Label

The labels, which are considered mutually independent, conditional only to C, are supplied by the dataset. From the latter, the probability of each label can be calculated according to the formula presented in equation 3.11.

The result is dependant on the chosen user since N_{L_k} represents the number of times L_k occurs in their routine, and N_L defines the total number of labels in that same routine.

$$P(L_k) = \frac{N_{L_k}}{N_L} \tag{3.11}$$

3.5.2.C Final Response Probability Model

Now with the values of $P(A \mid L)$, $P(\overline{A} \mid L)$ and $P(L)$ known for each label, the before detailed formulas can be applied in order to obtain $P(L \mid C)$ and $P(\overline{L} \mid C)$. As previously stated, there is an

additional element used to represent user discontentment with notification volume, β. That element is incorporated in equations 3.12 and 3.13, finalizing our expressions as,

$$P(A \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \propto \beta \left[P(A) \prod_{j=0}^{i} [P(L_j \mid A)] * \prod_{k=i+1}^{l_t} \left[P(\overline{L_k} \mid A)\right] \right] \tag{3.12}$$

$$P(\overline{A} \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \propto (1-\beta) \left[P(\overline{A}) \prod_{j=0}^{i} \left[P(L_j \mid \overline{A})\right] * \prod_{k=i+1}^{l_t} \left[P(\overline{L_k} \mid \overline{A})\right] \right] \tag{3.13}$$

For each instance, the above presented factors are estimated, normalized, and, resorting to a simple sampling method (equations 3.14 and 3.15), the simulator's response is determined. We believe this sampling technique allows us to reflect the ambiguity of users more accurately.

$$X \sim U(0,1) \tag{3.14}$$

$$\hat{c} \in \left\{A, \overline{A}\right\} \sim \left\| P(C \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \right\| \leqslant X \quad C \in \left\{A, \overline{A}\right\} \tag{3.15}$$

In equation 3.15, $\hat{c}$ represents the class that defines the simulator's response to the notification.

Depending on the simulator's reaction and the state of the environment, the algorithm then obtains the respective reward and adjusts the strategy accordingly.

4

Experiments

Contents

This chapter aims to describe the experiments performed to test the previously detailed approach and detail how these experiments should be evaluated.

4.1 Implementation - Software tools

A variety of supporting tools were employed to achieve what this thesis sets out to do. Below a brief description of the libraries applied throughout this work is presented.´

- Random - Randomness is an essential component of ML algorithms' implementation, configuration, and evaluation. Hence, this work resorts to Random [73], a Python module that implements pseudo-random number generators for various distributions. This module utilizes a popular, robust, and thoroughly tested generator called the Mersenne Twister [74].

- Numpy - Numpy [75] is an open-source library developed for Python that aims to facilitate numerical computing. It does so by providing support for hefty, multi-dimensional matrices and arrays, as well as a collection of mathematical operations users can execute on these data structures. It is a fundamental package for scientific computing that strives for code optimization by removing the need for explicit indexing or looping. The use of vectorization transforms the code into a more simple, powerful, and efficient approach.

- Math - The Math module [76] comes packaged with Python, requiring no further installation. It enables the usage of the mathematical operations defined by the C standard for non-complex numbers. A set of methods and constants allow for advanced operations like exponential, logarithmic, trigonometric, and power functions. Although other Python libraries such as Numpy have similar purposes, Math is better suited for scalar values, offering a faster and more efficient performance. Excluding a few functions, the Math module is, however, incapable of dealing with arrays.

- Pandas - This open-source Python software package [77] expedites data manipulation and analysis by offering a set of functional data structures and operations for manipulating numerical tables and time series.

- Tensorflow - TensorFlow [78] is an open-source software released by Google in 2015 that enables ML and Artificial Intelligence. In this work, it is used due to its particular simplification of the training and inference of RL-based deep NN. This package is more complex than the formerly mentioned, providing a comprehensive and flexible ecosystem of tools, libraries, and resources. Moreover, it is in constant evolution, supporting the effortless deployment of ML models in the cloud, on-prem, on a browser, or on-device, independently of the language used.

In 2017, Google released TensorFlow Lite. Specific for mobile development, it permits the conversion of regular TensorFlow models into compressed ones that work more efficiently in mobile or embedded devices. Although TensorFlow Lite is not used in this thesis, TensorFlow is here applied with the aim that, in the future, this conversion can be made, making our work more efficient when deployed on mobile devices.

- Gym - Gym [79] is an open-source toolkit specifically designed to support RL algorithms' development, comparison, and evaluation. This library offers a vast collection of pre-built environments that ordinary users can quickly set up and test their algorithms on. Furthermore, it allows the creation of any desired custom environment, letting users tailor their observation and action spaces, as well as reward signals. As an added benefit, this package is compatible with any numerical computation library (such as TensorFlow) that developers may use to implement their RL agents.

  In this work, Gym was utilized to design and implement our custom environment, connecting the algorithms and the simulated user.

4.2 Model Initialization Methods

One of the main objectives of this thesis is to analyze the various efficiency levels that algorithms can obtain in our context when the utilized models are initialized in different manners. Hence, three methods were chosen and experimented with; results on this subject are further discussed in the following chapter.

4.2.1 No Previous Knowledge Models - Online Learning

As a simplistic approach, the first selected method entails no previous experience, implying that the models start with no prior knowledge and learn only from interaction with a specific user. It is expected that this method would allow for better customization of our models, providing better results in the long run. However, it is also anticipated that it shall take longer to start doing so since it must learn every user preference from scratch.

As these models never cease learning, if users change their routine at any time, this model would be able to adapt to those changes.

4.2.2 Previously trained models - Offline Learning

In this approach, models are trained with two different users before being tested with a third one, where they only apply what they have learned from previous experience. These models are expected to provide acceptable results from the moment they are deployed since they are likely to have already

learned generic preferences that people share amongst different lifestyles. Nonetheless, since no learning happens after the training stage, models are not personalized and are not expected to reach such good results as the remaining methods. Furthermore, and for that same reason, these models will not efficiently adapt to routine changes that users may go through.

4.2.3 Previously trained adaptive models - Combination of Offline and Online Learning

In this case, models are likewise trained with two users before being deployed. However, they continue learning, which allows them to start more efficiently than models with no previous knowledge while also growing to be customizable. Assuming the chosen users' routines are varied enough to provide generalized knowledge that could then be applied to any user, this model, which combines the two previous ones, is expected to offer the best and most stable results.

4.2.4 Cross-validation

Two of the most commonly applied cross-validation methods are N-fold and Leave-one-out [26]. Both consist of splitting one dataset into training and validation sets, differing only on the splitting process.

In N-fold, random splits can be executed regardless of the participants or the sequentiality of their data. In leave-one-out, models are trained with all users except one, which is then used to test the model's performance. Although the former might imply using a small number of samples when the quantity of available participants is reduced, it respects the sequentiality of the data.

In problems such as ours, where the wish is to demonstrate that models can learn when users are more available in their routines, the order of the data is relevant. Hence, leave-one-out was the method utilized to implement the initialization methods Previously Trained and Previously Trained Adaptive. In the next section, further detail on the significance of this sequentiality is given.

4.3 Users' Routine - Daily vs. weekly routine

This thesis hypothesizes that patterns of opportune timings can be found in a person's routine. It is known that, for most people, activity patterns follow a week-based rhythm, for instance, working from 9h to 18h on weekdays or going to the gym on consistent days. Furthermore, on weekends, people tend to possess more irregular routines and have more free time, which usually implies more availability for notifications [43]. When analyzing days independently from each other, these repetitive patterns would not be as easily detected. The model would learn how a generic day is for the user but not the difference between their schedule on a Monday or on a Saturday.

By applying the different state representations of Q-learning and DQN, described in Section 3.3, and the different formulations of UCB, described in Section 3.2.6, this work intends to test if higher levels of efficiency can be obtained when letting the agent learn what a typical week is for the user (UCB Week, S1, S2) instead of a typical day (UCB Day, S3, S4). It is expected that when modeling opportune timings throughout a week, the agent takes longer to learn, but if enough time is provided, better results can be obtained. It is essential that it can be understood if this extra time taken to train provides improvements that are considered significant enough for the user to accept the disturbance of a not yet so well-trained model for more extended initial periods.

4.4 Performance Metrics

As performance metrics of our algorithms, we selected two: Goal Achievement Rate (G_r) and Notification Volume (N_v). Goal achievement rate, in equation 4.1, is defined as the fraction of accepted notifications (N_A) over the number of episodes being tested (N_{Days}, each episode representing a day).

$$G_r = \frac{N_A}{N_{Days}} \tag{4.1}$$

High G_r values show that our agent was able to identify when users are open to receiving and answering notifications throughout the day. However, an agent may increase the goal achievement rate by simply increasing interaction with users. Thus, the volume of sent notifications is also tracked to balance this effect. With equation 4.2, the average number of notifications sent per day, also referred to as notification volume, can be obtained.

$$\begin{aligned} N_v &= \frac{N_{Sent}}{N_{Days}} \\ &= \frac{\sum_{i=0}^{Days}(N_{A_i} + N_{\overline{A}_i})}{N_{Days}} \end{aligned} \tag{4.2}$$

A well-behaved agent presents a high response rate (G_r) while maintaining a low notification volume (N_v), ensuring in this way that our system gets a response without bothering the user when he is not receptive.

5

Simulation Results and Discussion

Contents

This chapter entails the presentation and discussion of the outcomes of the described simulations. Obtained results are compared, resorting to the metrics detailed in Section 4.4, and the best combination for each initialization method is discussed. Lastly, the best combination amongst all our permutable variables (initialization method, algorithm, state, and reward) is analyzed. In this final chapter, the impact of rewards and state representations is also discussed in further detail.

In all the tables shown throughout this chapter, the average G_r and N_v values obtained over 300 days of training are presented. When values are highlighted, that signifies that those represent the best performing results of the respective section. Furthermore, these are highlighted with the color with which they are then portrayed in a more detailed figure, which shows the average result among tested users, employing the Notification volume (N_v) and Goal Achievement Rate (G_r) metrics. The standard deviation was also analyzed and is likewise depicted in the displayed graphs.

Ideally, the notification system should have the following properties:

- G_r approaches 100%;
- N_v approaches 1 notification per day;
- Standard deviation values should not be high, ensuring consistency when applying the models to different users and in various situations;
- The obtained results are stable, meaning that the provided service should be consistent, not showing significant, unexpected variations in the N_v and G_r values throughout the training process. Although the system is expected to explore continually, user experience should not be affected in such a way that users would start experiencing dissatisfaction towards the application;
- From the moment of deployment, on day 1, the agent should not require a long period of time to start providing an acceptable level of user experience. Users must be able to start seeing the benefits of their effort early on, reinforcing user engagement and commitment.

5.1 Discussion - Initialization methods

Experiments with the developed simulator were performed for all possible combinations of the described initialization methods, algorithms, state representations, and rewards. Three tests were executed for all permutations by applying the leave-one-out technique for a set of 3 simulated users. The respective median was then determined as a measure of central tendency to diminish the influence of possible outliers.

As detailed in Chapter 4 we tested three main scenarios:

- A pre-trained model that does not utilize online user feedback (Previously Trained);

- A pre-trained model that also takes into consideration continuous online user feedback (Previously Trained Adaptive);
- A fully online trained model that learns continuously from user feedback, starting with no prior knowledge (No Previous Knowledge).

5.1.1 Previously trained model

UCB	Reward	G_r	N_v
Day	R1	0.993±0.008	3.174±1.206
	R3	0.981±0.023	1.882±0.141
	R5	0.985±0.008	1.996±0.257
	R6	0.814±0.192	1.402±0.257
Week	R1	0.969±0.025	3.316±1.051
	R3	0.970±0.014	2.254±0.755
	R5	0.929±0.047	2.139±0.492
	R6	0.772±0.079	1.417±0.125

Table 5.1: Previously Trained - UCB.

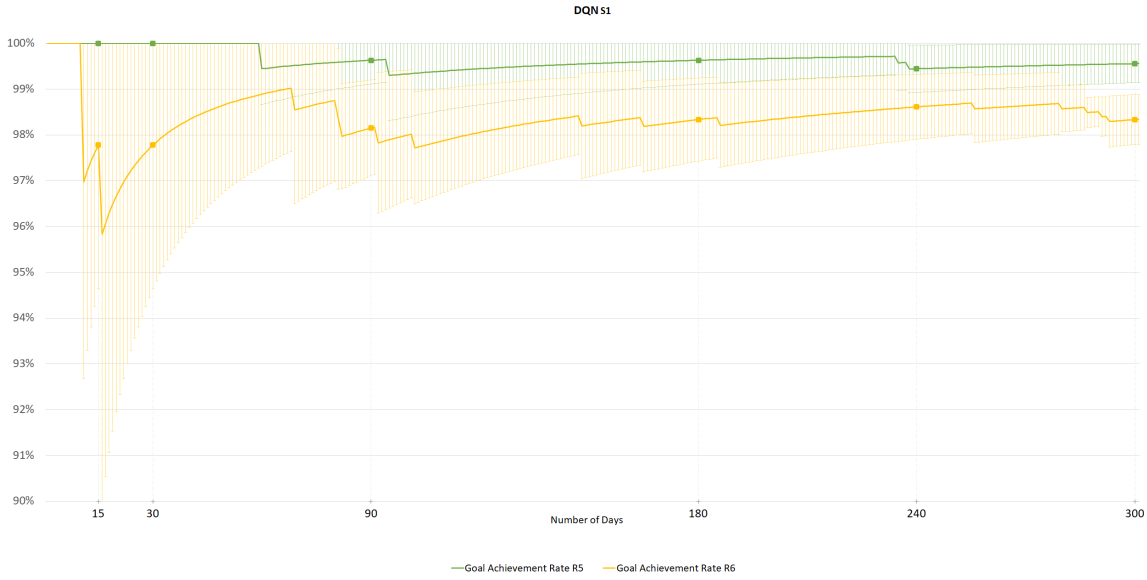
TQ	Reward	G_r	N_v	**TQ**	Reward	G_r	N_v
S1	R1	0.999±0.001	4.415±1.585	S3	R1	0.999±0.001	4.541±2.078
	R3	0.999±0.001	3.607±1.164		R3	0.993±0.005	3.487±1.154
	R5	0.989±0.005	2.304±0.332		R5	0.977±0.014	2.464±0.735
	R6	0.980±0.011	2.304±0.558		R6	0.966±0.025	2.113±0.180
S2	R1	0.918±0.086	6.387±3.255	S4	R1	0.911±0.109	6.150±4.238
	R3	0.933±0.066	5.747±2.637		R3	0.982±0.012	3.318±0.929
	R5	0.959±0.034	3.796±1.945		R5	0.949±0.025	1.877±0.025
	R6	0.979±0.007	3.219±1.002		R6	0.931±0.009	2.167±0.231

Table 5.2: Previously Trained - Tabular Q-learning.

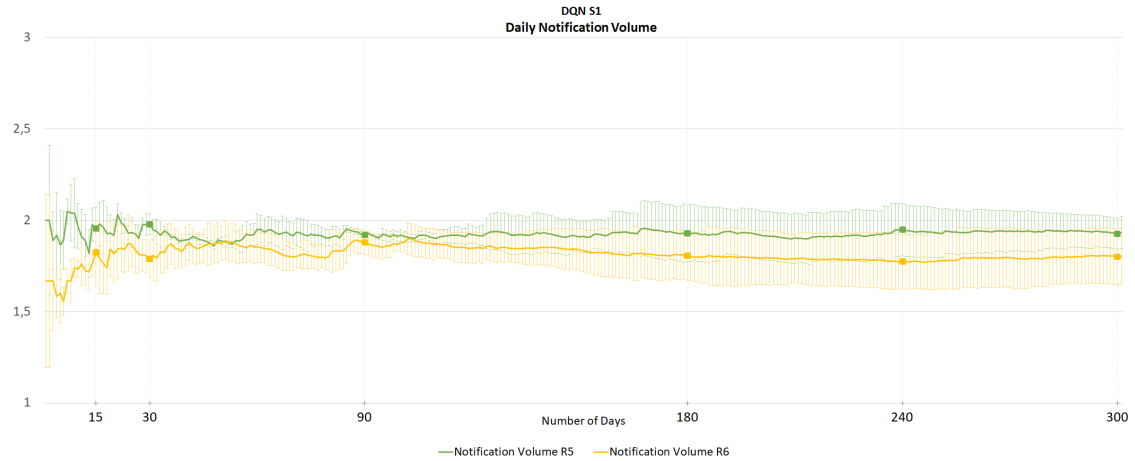
DQN	Reward	G_r	N_v	**DQN**	Reward	G_r	N_v
S1	R1	1±0	5.350±2.377	S3	R1	0.755±0.346	3.444±2.879
	R3	1±0	3.271±0.955		R3	0.732±0.378	3.410±2.806
	R5	0.996±0.005	1.926±0.095		R5	0.996±0.004	1.931±0.082
	R6	0.983±0.012	1.813±0.123		R6	0.992±0.003	1.864±0.121
S2	R1	0.886±0.082	8.273±4.251	S4	R1	0.899±0.074	7.999±4.092
	R3	0.928±0.077	8.180±4.117		R3	0.924±0.090	7.039±4.002
	R5	0.326±0.186	0.650±0.124		R5	0.485±0.366	0.927±0.366
	R6	0.461±0.296	1.902±0.998		R6	0.255±0.058	0.568±0.046

Table 5.3: Previously Trained - DQN.

By analyzing Tables 5.1 to 5.3, it is visible that more generic state representations generate superior results in this initialization technique. Thus proving that if the purpose is to learn generic preferences, it should be done in the less detailed manner possible, which in this context is represented by modeling a nonspecific daily routine (S3).

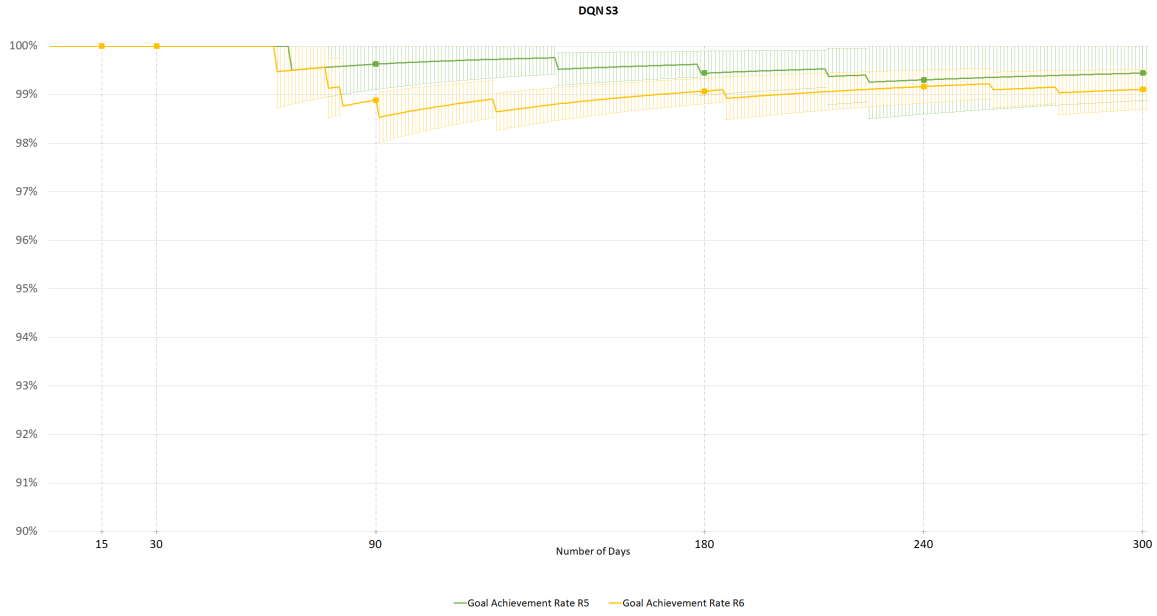


(a) Goal Achievement Rate

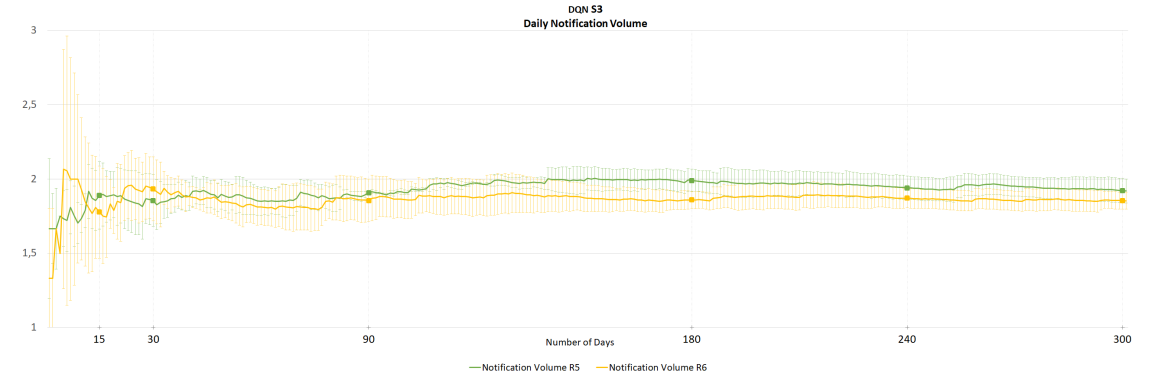


(b) Notification Volume

Figure 5.1: Previously Trained, DQN, State S1 - average result among tested users over 300 days of training.



(a) Goal Achievement Rate



(b) Notification Volume

Figure 5.2: Previously Trained, DQN, State S3 - average result among tested users over 300 days of training.

When applying DQN, Figures 5.1 and 5.2, the network is able to employ the less complex state representations (S1, S3) and leverage them throughout the training phase, learning generic user preferences better than any other combination. With this learning, although not adaptable to new users' routines, a consistently pleasant user experience can still be offered, seeing as a reasonable G_r is obtained while the average N_v is kept below two. However, if applied to a user that has atypical habits, this model would not prove satisfactory since, at its core, it is not learning specific user preferences and adapting to their schedule, but simply applying previous knowledge.

Furthermore, as one can perceive in the tables presented throughout this section, when compared to the Previously Trained Adaptive or the No Previous Knowledge methods, the Previously Trained model here presented displays overall higher standard deviation levels. These represent a wider dispersion of the data concerning the presented mean, indicating that the values gathered for each user are not as consistent as the ones obtained in the remaining simulations.

These simulations confirmed that a generic prediction model trained on multiple users' data tends to be less accurate for predicting the interruptibility of a specific user than individual-based models. This statement can be confirmed in Table 5.10, where the average values obtained for each initialization method are stated. Nonetheless, this method obtains satisfactory results when resorting to the right combination of algorithm, state, and reward structure. Although relevant, and a possible solution for this thesis' challenges, this implementation highly resembles intelligent notification systems that apply supervised learning techniques, not fully exploring the potential benefits of reinforcement learning.

5.1.2 Previously trained adaptive model

This technique implements an online learning method bootstrapped with a model built on previous knowledge. As expected, and shown in Tables 5.4 to 5.6, and Table 5.10, this method appears to provide the overall best performing results amongst all three initialization techniques since it can be refined as the final user is actively using the application.

Both UCB and Q-learning present a good balance between G_r and N_v. Nonetheless, due to the previous knowledge obtained from the training, results may present higher variation values at the beginning of their deployment, depending on the user and how well they fit into the previously learned generic lifestyle. With time, as the models learn the specific routine of the user in question, the mentioned variation would decrease. This variation can be seen in Figures 5.3 to 5.5 and in Tables 5.4 to 5.6 where the standard deviation values are presented. Furthermore, although N_v converges early on, G_r values do not, which signifies that the agent is still exploring and attempting to learn the user's most opportune timings. While doing so, and due to the consequent inconsistency in the provided notification service, this behavior may put at risk user engagement with the application.

UCB	Reward	G_r	N_v
Day	R1	0.998±0.005	2.555±0.488
	R3	0.987±0.014	1.721±0.127
	R5	0.987±0.009	1.716±0.009
	R6	0.752±0.169	1.234±0.252
Week	R1	0.984±0.014	2.656±0.696
	R3	0.953±0.038	1.914±0.397
	R5	0.872±0.029	1.578±0.279
	R6	0.828±0.128	1.285±0.191

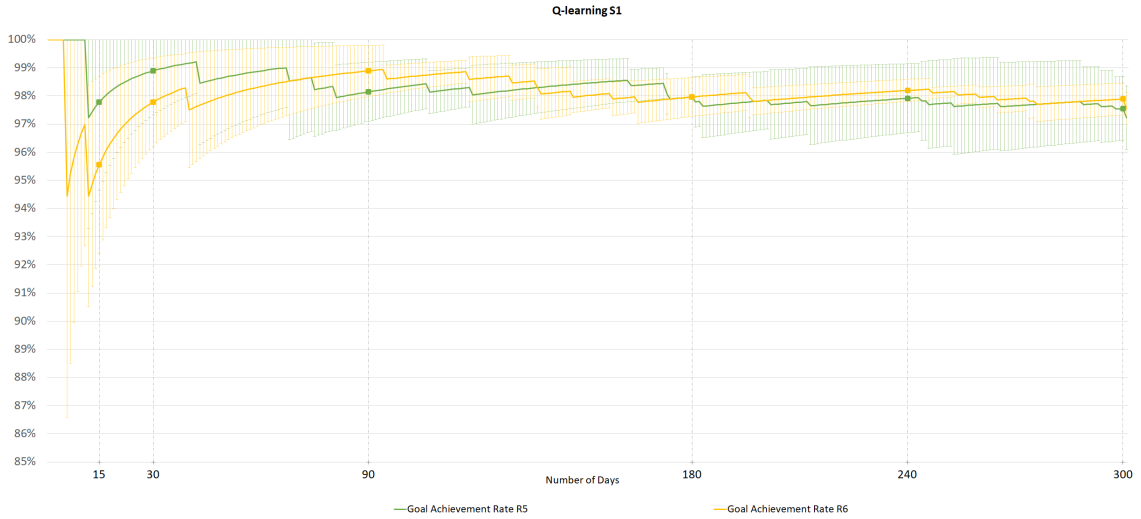
Table 5.4: Previously Trained Adaptive - UCB.

TQ	Reward	G_r	N_v	TQ	Reward	G_r	N_v
S1	R1	0.997±0.005	4.090±1.426	S3	R1	0.998±0.002	4.272±1.487
	R3	0.994±0.005	3.026±0.813		R3	0.996±0.006	2.705±0.629
	R5	0.982±0.013	1.907±0.289		R5	0.942±0.011	1.849±0.014
	R6	0.981±0.009	1.949±0.237		R6	0.929±0.020	1.776±0.123
S2	R1	0.962±0.017	3.367±1.073	S4	R1	0.967±0.007	3.064±0.591
	R3	0.985±0.009	2.455±0.581		R3	0.979±0.009	2.339±0.298
	R5	0.985±0.004	1.945±0.285		R5	0.963±0.017	1.935±0.215
	R6	0.981±0.011	1.922±0.298		R6	0.964±0.017	1.839±0.108

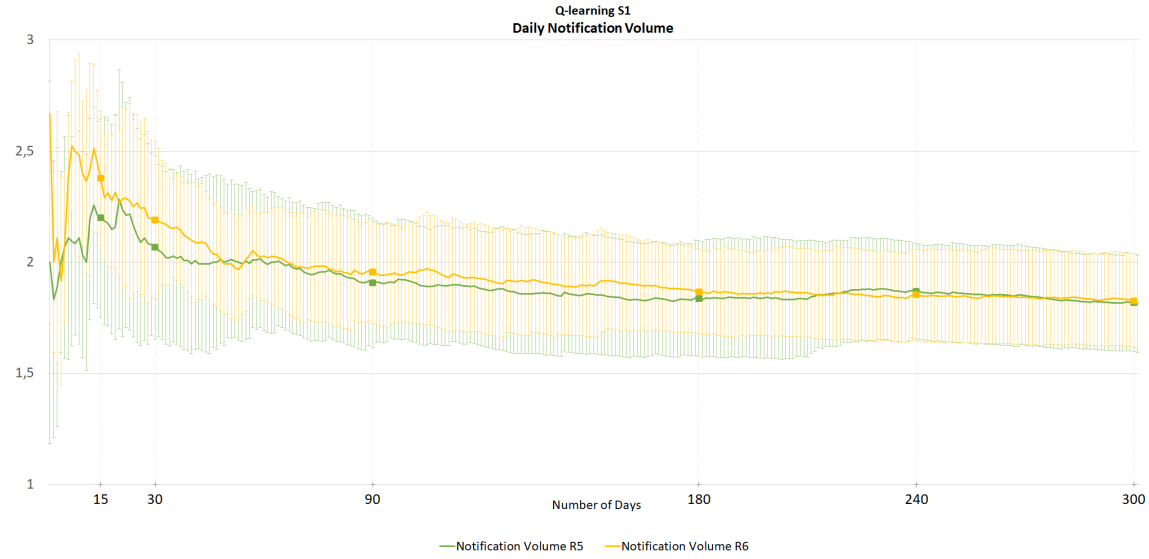
Table 5.5: Previously Trained Adaptive - Tabular Q-learning.

DQN	Reward	G_r	N_v	DQN	Reward	G_r	N_v
S1	R1	0.984±0.055	3.769±2.117	S3	R1	0.979±0.049	3.721±1.615
	R3	0.982±0.077	3.905±1.547		R3	0.983±0.216	3.656±1.301
	R5	0.971±0.208	3.372±1.414		R5	0.974±0.054	3.396±1.051
	R6	0.976±0.033	3.309±1.256		R6	0.976±0.027	3.441±1.188
S2	R1	0.958±0.069	4.623±2.299	S4	R1	0.957±0.088	4.468±2.126
	R3	0.978±0.074	3.949±1.487		R3	0.965±0.081	4.129±1.419
	R5	0.964±0.012	3.609±1.284		R5	0.979±0.007	3.533±1.910
	R6	0.961±0.010	3.586±1.265		R6	0.969±0.013	3.516±1.051

Table 5.6: Previously Trained Adaptive - DQN.

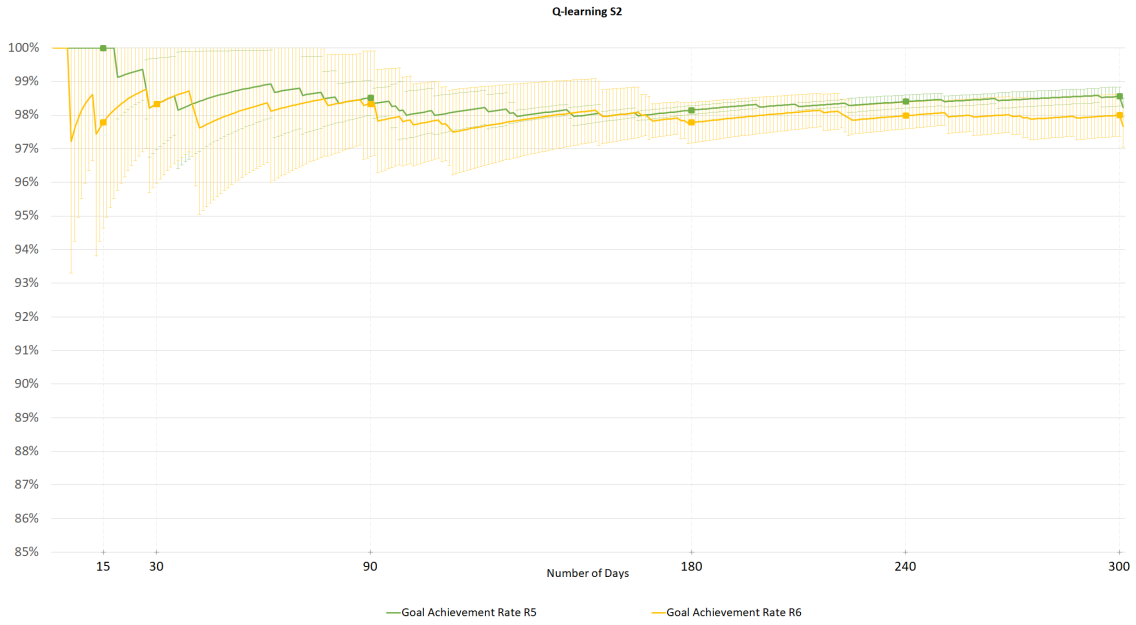


(a) Goal Achievement Rate

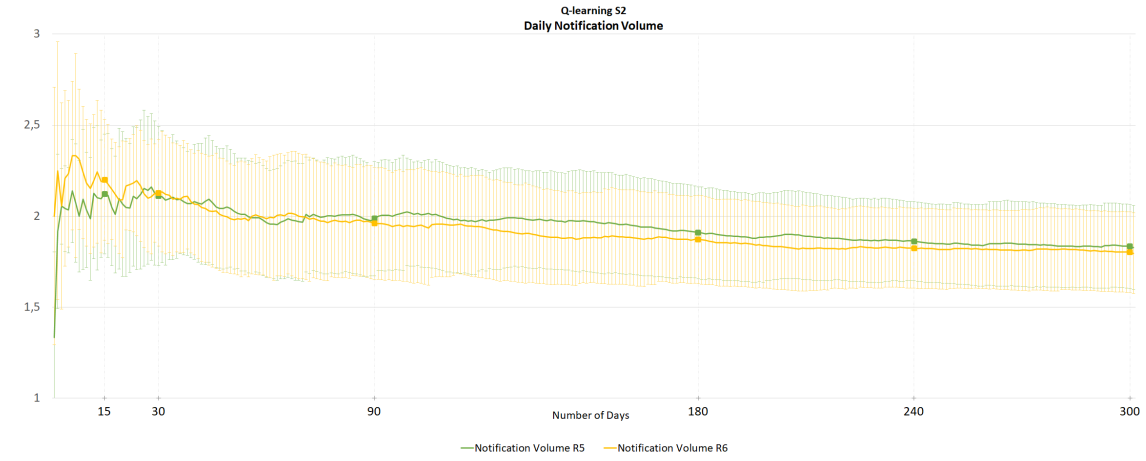


(b) Notification Volume

Figure 5.3: Previously Trained Adaptive, Tabular Q-learning, State S1 - average result among tested users over 300 days of training.

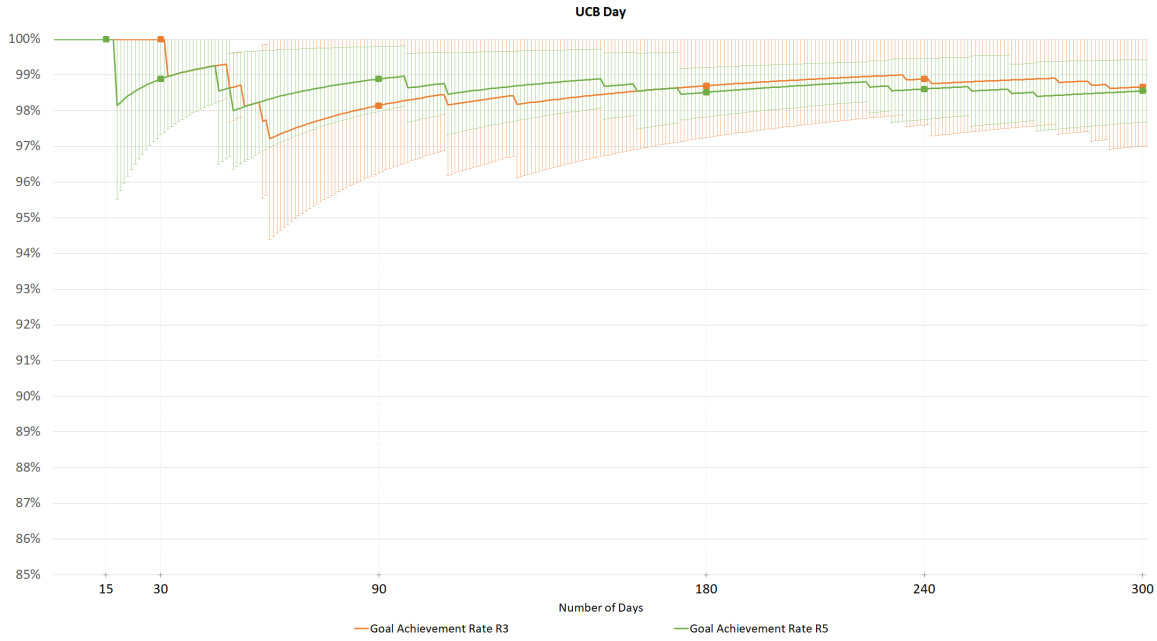


(a) Goal Achievement Rate

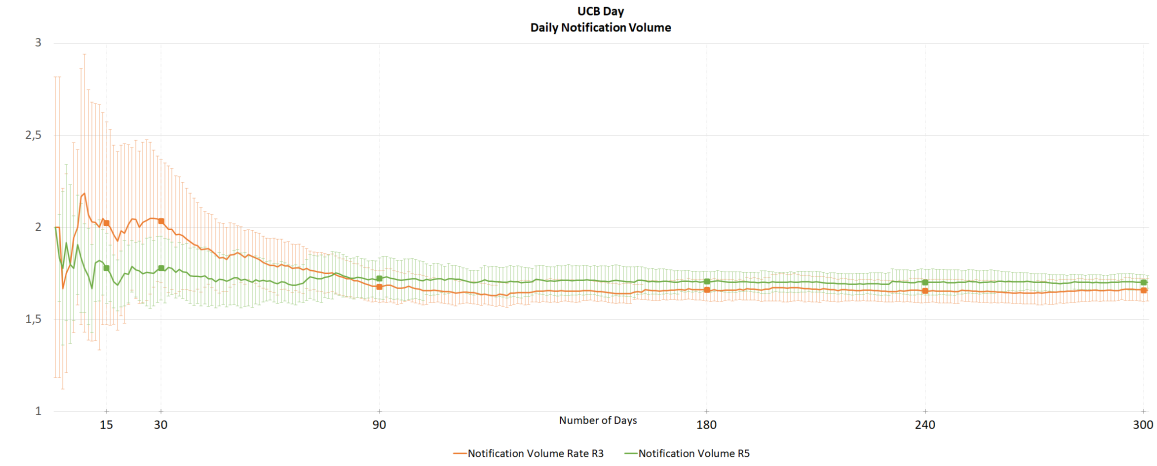


(b) Notification Volume

Figure 5.4: Previously Trained Adaptive, Tabular Q-learning, State S2 - average result among tested users over 300 days of training.



(a) Goal Achievement Rate



(b) Notification Volume

Figure 5.5: Previously Trained Adaptive, UCB Day - average result among tested users over 300 days of training.

5.1.3 No Previous Knowledge

UCB	Reward	G_r	N_v
Day	R1	0.960 ± 0.005	2.810±0.374
	R3	0.903±0.081	1.811±0.181
	R5	0.979±2.8∗10^{-17}	1.714±2.8∗10^{-17}
	R6	0.973±0.004	1.529±0.092
Week	R1	0.804±0.052	3.959±1.322
	R3	0.846±0.028	2.332±0.520
	R5	0.754±0.045	1.916±0.351
	R6	0.717±0.153	1.656±0.247

Table 5.7: No Previous Knowledge - UCB.

TQ	Reward	G_r	N_v	TQ	Reward	G_r	N_v
S1	R1	0.999±0.001	3.876±1.113	S3	R1	0.999±0	4.098±1.267
	R3	0.998±0.002	3.367±1.008		R3	0.999±0	3.015±0.613
	R5	0.999±0	2.699±0.633		R5	0.977±0.012	2.112±0.211
	R6	0.999±0	2.684±0.660		R6	0.972±0.015	2.064±0.211
S2	R1	0.970±0.019	3.222±1.002	S4	R1	0.970±0.014	3.142±0.518
	R3	0.977±0.014	2.843±0.734		R3	0.988±0.004	2.394±0.345
	R5	0.972±0.009	2.472±0.539		R5	0.971±0.015	2.195±0.015
	R6	0.984±0.009	2.412±0.548		R6	0.966±0010	2.271±0.220

Table 5.8: No Previous Knowledge - Tabular Q-learning.

DQN	Reward	G_r	N_v	DQN	Reward	G_r	N_v
S1	R1	0.930±0.007	4.951±2.006	S3	R1	0.952±0.009	4.467±1.603
	R3	0.818±0.007	4.036±1.387		R3	0.696±0.003	2.556±1.140
	R5	0.448±0.010	0.968±0.060		R5	0.536±0.007	1.202±0.007
	R6	0.449±0.004	1.115±0.128		R6	0.521±0.011	1.203±0.091
S2	R1	0.882±0.016	8.495±3.730	S4	R1	0.890±0.016	8.123±3.762
	R3	0.915±0.012	7.545±2.173		R3	0.909±0.016	7.501±1.966
	R5	0.968±0.015	1.803±0.177		R5	0.979±0.006	1.810±0.035
	R6	0.964±0.012	1.917±0.185		R6	0.958±0.017	1.679±0.095

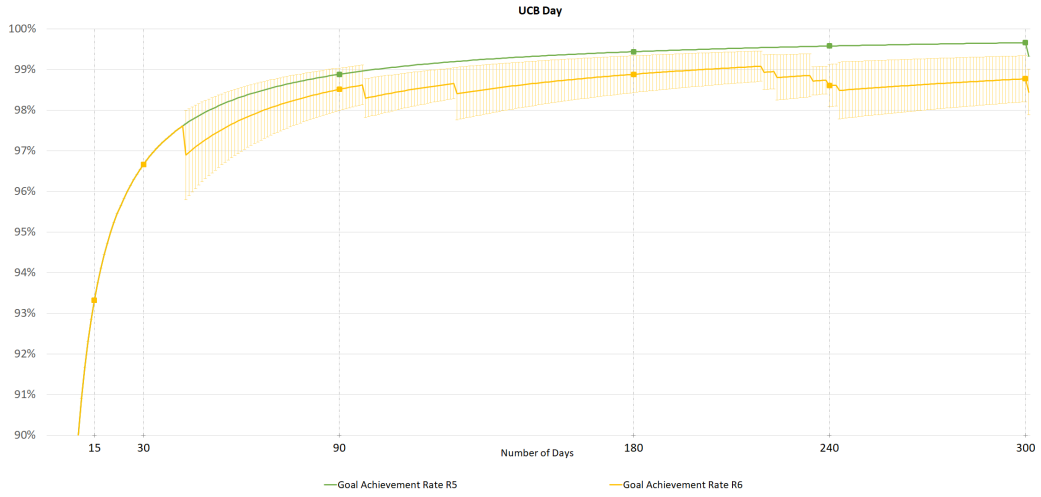
Table 5.9: No Previous Knowledge - DQN.

Looking just at Tables 5.7 to 5.9 and Table 5.10, this method appears to be overall less efficient than the Previously Trained Adaptive model, taking longer to achieve efficient G_r values (2 to 3 months). When analyzing Figures 5.6 and 5.7 in more detail, it can be concluded that although the average values are worse than the ones detailed in the other initialization methods, that is caused by the model's lack of previous experience and knowledge. Meaning that in the beginning this model provides a worse user experience, and G_r and N_v results, due to the fact that it is learning every user preference from scratch. These initial results diminish the presented average G_r values while raising the average N_v values. However, in the long run, the personalized models achieve better results that converge faster than the ones shown in the Previously Trained Adaptive model. This convergence implies that users are

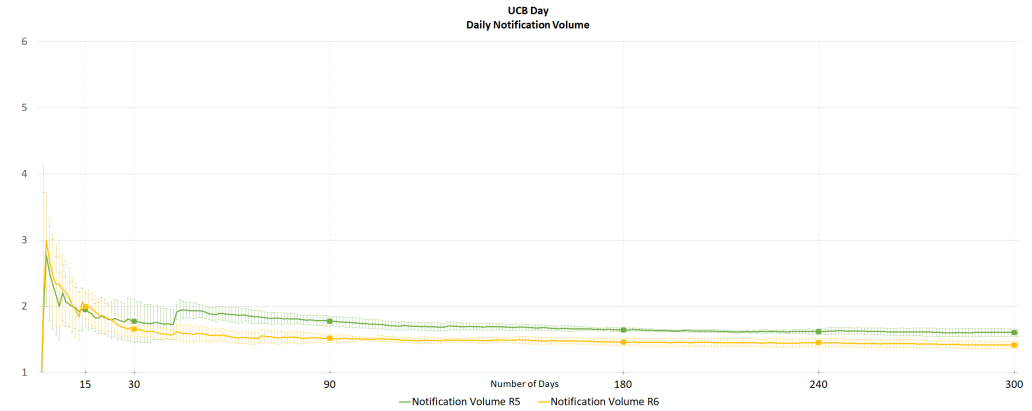
provided with a more consistent experience due to the system's continued success at finding opportune times, hence maintaining a high G_r, and the lack of peaks in the N_v values, consistently and steadily approaching about two notifications per day as time goes by.

It should be noted that, comparing these results with all other initialization methods, here the analyzed metrics do not deteriorate, helping to ensure that users do not feel disappointed with the system's promise to be personalized.

For this case, simple combinations of both UCB and the Q-learning algorithm, with state representation S1, provide the best results. Furthermore, the overall lower deviation values were obtained with this implementation, as visible in Table 5.10

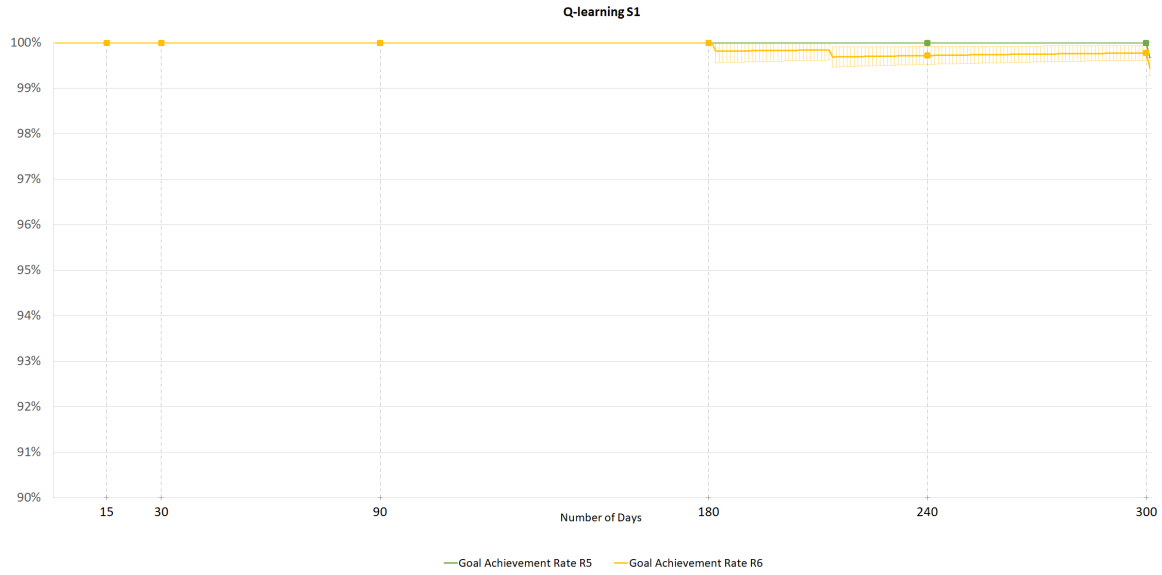


(a) Goal Achievement Rate

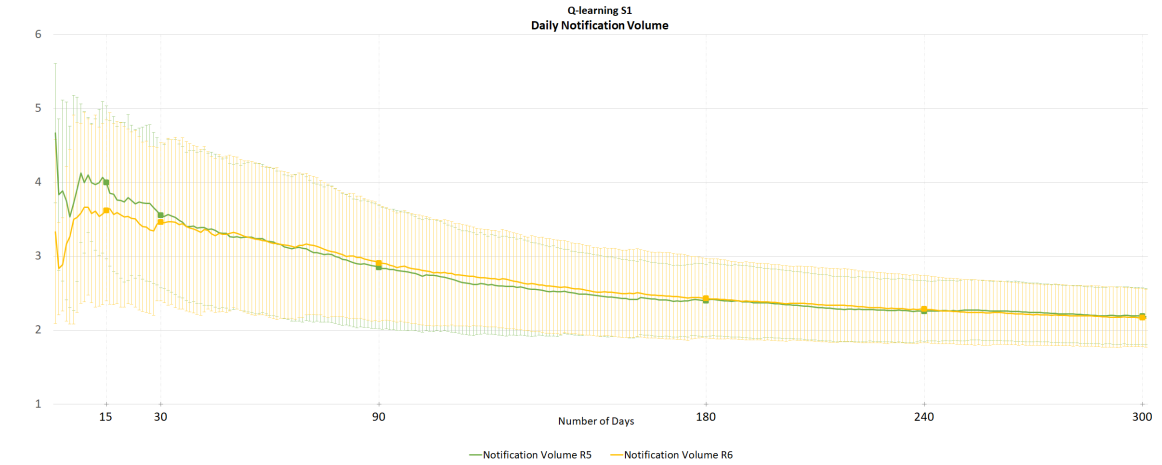


(b) Notification Volume

Figure 5.6: No Previous Knowledge, UCB Day - average result among tested users over 300 days of training.



(a) Goal Achievement Rate



(b) Notification Volume

Figure 5.7: No Previous Knowledge, Tabular Q-learning, State S1 - average result among tested users over 300 days of training.

5.2 Overall Results

With the aim of providing a bit more clarity regarding the overall conclusions that can be reached from the obtained results, the average values over all simulations of all the combinations of algorithms, state representations and reward structures in each initialization method (Table 5.10) and of initialization methods, state representations and reward structures in each algorithm (Table 5.11) are now shown.

———	G_r	N_v
No Previous Knowledge	0.887±0.017	3.049±0.785
Previously Trained	0.888±0.069	3.351±1.338
Previously Trained Adaptive	0.963±0.041	2.877±0.881

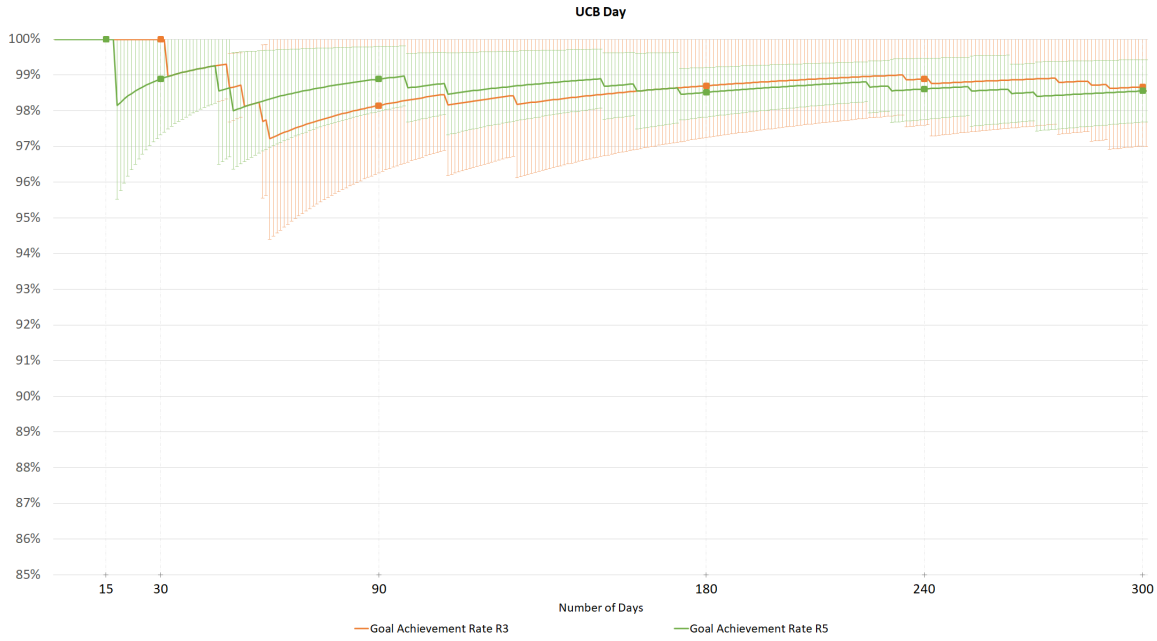
Table 5.10: Average values over all combinations of each initialization method.

———	G_r	N_v
UCB	0.905±0.049	2.082±0.399
TQ	0.975±0.015	2.983±0.837
DQN	0.854±0.067	3.706±1.467

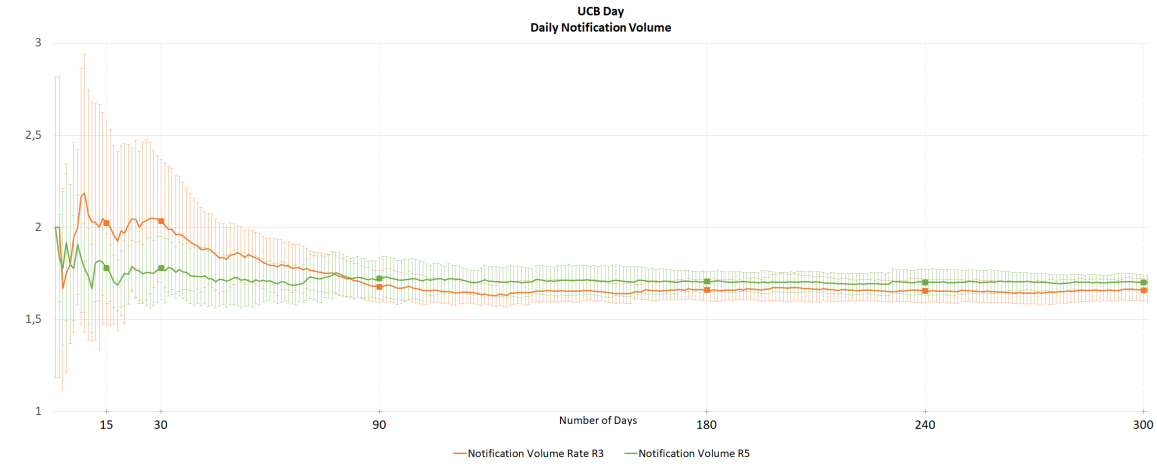
Table 5.11: Average values over all combinations of each algorithm.

5.2.1 Best Combination

Amongst the obtained results, the best performing combination, shown in ComboA - Figure 5.8, utilized UCB Day, for which a state representation is not required. As already detailed in Section 3.2.6, UCB Day applies an upper confidence bound algorithm to each hour of the day, helping the model learn what action is better for each moment.



(a) Goal Achievement Rate



(b) Notification Volume

Figure 5.8: ComboA: Previously Trained Adaptive, UCB Day - average result among tested users over 300 days of training.

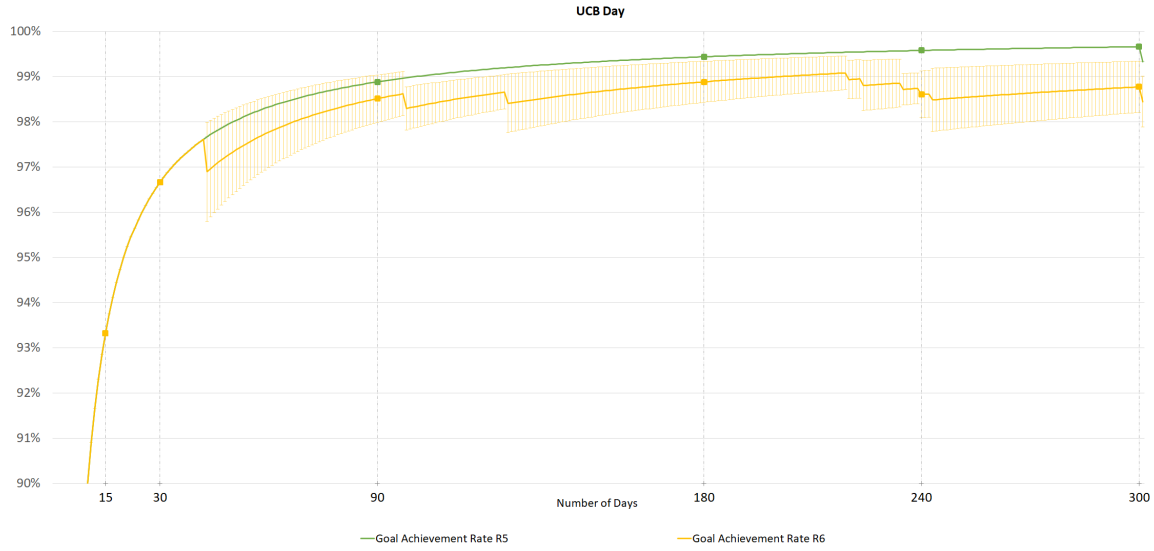
The initialization method applied in this case was the Previously Trained Adaptive, providing a high G_r rate from the model's deployment while still allowing it to adapt over time to the user's specific routine, consequently achieving a lower N_v as time goes on. Additionally, this combo implemented reward R5, which proved to offer the best balance between receiving an answer and not bothering the user with many notifications throughout the day.

This same combo but with reward R3 provided analogous results. In contrast, R6 produced a worse user experience due to the higher penalization value for unanswered notifications, leading to fewer notifications sent but also to a lower G_r.

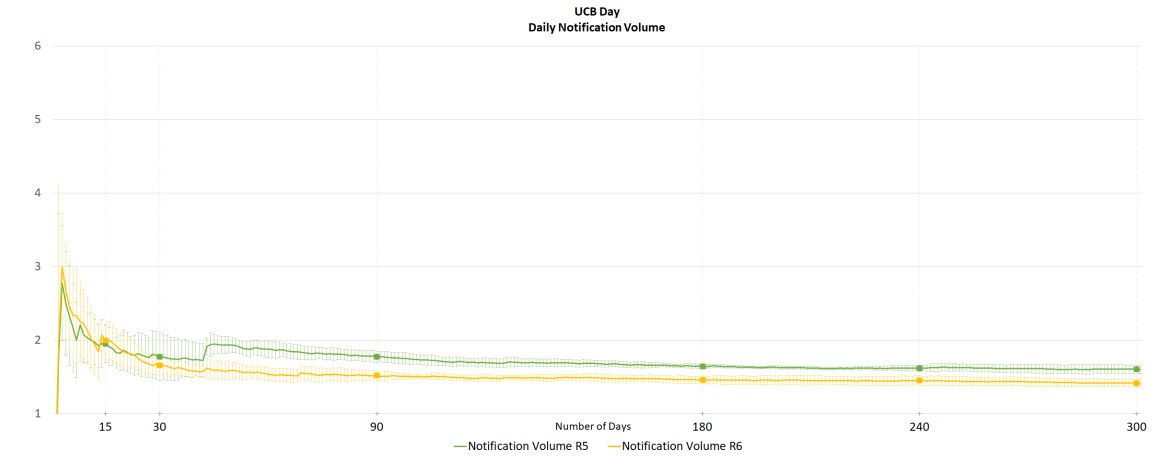
Similar results, visible in ComboB - Figure 5.9, were achieved in the best combination of the No Previous Knowledge method, also resorting to UCB Day and reward R5. However, ComboB's initialization method implies starting with no previous knowledge of generic user preferences, leading to initially lower G_r and slightly higher early N_v values. It takes approximately two months to achieve G_r values equivalent to the ones obtained with ComboA.

Contrarily to ComboA, here, the reward that offered a comparable outcome utilized reward R6, demonstrating that, when starting with no former experience, punishing the agent more severely for sending notifications that go unanswered generates a superior performance.

It is relevant to note that although not providing the best performance, the combination that offered acceptable results consistently throughout all initialization options was implemented with Tabular Q-learning, state S1, and rewards R5 and R6. This affirmation is supported by Tables 5.2, 5.5 and 5.8 and Table 5.11.



(a) Goal Achievement Rate



(b) Notification Volume

Figure 5.9: ComboB: No Previous Knowledge, UCB Day - average result among tested users over 300 days of training.

DQN presented the most unstable and inconsistent effects, which is reflected both in Tables 5.3, 5.6 and 5.9 and Table 5.11, providing an overall less pleasant user experience. Nonetheless, when applied in the Previously Trained method, it provided the best results for that type of initialization. This topic was further developed in Section 5.1.1.

In a previously mentioned work, [36], the authors developed a similar system and achieved answer rates of about 89.6% with UCB and 93.8% with Q-learning, which improved pre-existent SL benchmarks of about 77.4%. This thesis concludes that better results can be offered by implementing simple RL algorithms, basic state representations, and readily available non-sensitive user information. Applying the best combination, ComboA, in this thesis we were able to obtain a response rate of 98.7%.

5.2.2 Best Performing State

There was not one individual state which commonly displayed a better performance, therefore, one cannot define the best performing state across all algorithms. This outcome was to be expected since the success of the implemented state representation is highly dependent on the utilized algorithm and respective computational requirements. Notwithstanding, it could be said that S1 and S3 tend to perform better throughout all combinations, revealing that more complex, detailed states are not necessarily always more efficient. This also implies that no conclusion can yet be taken on whether patterns of opportune timings are better learned for an average day (S3) or an average week (S1).

What can be confirmed is that, as long as appropriately designed, more straightforward states, such as S1 or S3, can contain all of the essential information for the problem, allowing for an efficient application of basic algorithms and providing superior results due to the consequent lower memory, time, and computational requisites.

5.2.3 Best Performing Reward

Contrarily to the state representation, the reward structure is highly linked and dependent on the problem and goal at hand instead of the applied algorithm. For this reason, it is possible to verify that the reward which generated a better overall performance in the tackled problem was R5, presented in Table 5.12.

Event \User Reaction	User does not answer	User answers	No reaction is possible
A notification is not sent	-	-	0
A notification is sent	-2	2	-
Episode ends without the goal being achieved (24h)	-3		

Table 5.12: Best overall reward structure - Reward 5.

An interruptibility management system using such a reward structure allows fewer notifications to be triggered while still providing a high enough penalty on the agent when sending notifications at inopportune timings or not achieving its final goal. Hence, these values provided the best balance between achieving one daily answered notification and not bothering the user, without keeping the agent from sending notifications altogether.

6

Conclusion

Contents

This chapter summarizes the work done and points out the system's shortcomings and possible future work that should be developed in order to enrich this project.

6.1 Conclusions

This work aimed to build an intelligent notification system that could adequately manage interruptions, created in a mHealth self-management application, by learning what moments were the most opportune for each user throughout their day. Theoretical background and related work were analyzed, and a detailed study of possible RL algorithms and world representations was developed. For the implementation of the proposed methodology, a set of algorithms and problem formulations (initialization method, state representation, and reward structure) were reviewed, selected, and tested, in an attempt to determine the most desirable combination of initialization, algorithm, state, and reward definition for the problem at hand.

Different users have different preferences regarding phone notifications and when or what they consider to be opportune timings in their routine. For that motive, and since acceptable results were obtained for a substantial amount of the tested permutations, it is challenging to select one as the best overall result. Nonetheless, a set of guidelines for the analysis of the results was introduced. Compared to previous works, the developed implementations generally offered more efficient performances, being able to achieve in their best combinations, answer rates of 98.7% in UCB, 99.9% in Tabular Q-learning, and 99.2% in DQN. These values were achieved always considering the objective of attaining one answer per day. For that purpose, UCB sent 1.7 notifications per day, Tabular Q-learning 2.7 per day, and DQN 1.9 per day. Although these values are very satisfactory, the evolution of the learning process highly influences the actual success of the notification system when applied with real users, hence why a more detailed discussion and analysis of the obtained graphs was required in order to assess the results. From that analysis, it was possible to determine that the overall best performing combination actually applied UCB Day with reward R5 and the Previously Trained Adaptive initialization method. It should also be noted that throughout our tests reward structure R5 consistently provided the best results, offering a good balance between achieving one daily answered notification and not bothering the user.

In conclusion, this work demonstrates that a balanced and efficient intelligent notification system can be built for the purpose of being applied to a mHealth application without requiring access to any private user information or device sensor. Hence, addressing some of the main concerns commonly shared among the targeted users, such as privacy topics or device and sensor-related issues and limitations.

6.2 System Limitations and Future Work

This thesis considers only two possible user reactions – immediately accept or immediately decline/ignore the received notification. Yet, the goal could be further detailed so that user reactions are not forced to be instant but can instead arrive within a predefined interval. This option would allow the user to have the chance to end or start a new activity, to which a higher receptivity level may be associated.

Furthermore, the available responses could be elaborated by introducing reactions previously proposed by other authors. For instance, in [80], McFarlane and Latorella suggested that there were 5 main possible user reactions to any interruption in human-computer relationships, from which 4 can be adapted to the problem this thesis addresses. Accepting a notification could be further divided into Preemptive or Intentional integration. The former represents an interruption that the user handles instantly before resuming its ongoing task. The latter is an interruption that, being relevant to the ongoing task, is integrated into it to be handled. Additionally, 2 more precise alternatives were provided in place of what was here considered as a simple decline: oblivious dismissal, where the notification goes unnoticed and intentional dismissal, where people explicitly decide not to address it.

Moreover, this study applies a simulated user, requiring further deployment and testing in actual mobile devices utilized by real users. To do so, TensorFlow Lite should be used, ensuring that our models are as efficient as possible. In addition, to fully ensure the validity and robustness of the proposed interruptibility management system, further testing should be carried out on users with different lifestyles, diseases, social contexts, and demographics. These would permit understanding which model is better applied to which specific type of user, allowing for even superior performance values to be obtained.

On an additional note, further research on different self-monitoring frequencies is necessary. In this work, it was considered that the user required only one self-management task in a day, hence why the established goal was one daily answered notification. However, having a deeper understanding of which frequency is the most appropriate and beneficial for the user's health would allow for a more precise goal and reward definition during the formulation of the RL problem, further ensuring the applicability of this work to a real-life context.

Bibliography

[1] Y. Vaizman, K. Ellis, and G. Lanckriet, "Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches," *IEEE Pervasive Computing*, vol. 16, no. 4, pp. 62–74, 2017.

[2] Geneva: World Health Organization, "Noncommunicable diseases progress monitor 2020," 2020.Licence: CC BY-NC-SA 3.0 IGO.

[3] United Nations Department of Economic and Social Affairs, Population Division, "World Population Ageing 2020 Highlights: Living arrangements of older persons," pp. 1–47, 2020.

[4] Geneva: World Health Organization, "The impact of the COVID-19 pandemic on noncommunicable disease resources and services: results of a rapid assessment," pp. 1–32, 2020.Licence: CC BY-NC-SA 3.0 IGO.

[5] Geneva: World Health Organization and the United Nations Development Programme, "Responding to non-communicable diseases during and beyond the COVID-19 pandemic: State of the evidence on COVID-19 and non-communicable diseases: a rapid review," 2020.Licence: CC BY-NC-SA 3.0 IGO.

[6] V. P. Cornet and R. J. Holden, "Systematic review of smartphone-based passive sensing for health and wellbeing," *Journal of Biomedical Informatics*, vol. 77, no. October 2017, pp. 120–132, 2018.

[7] N. Matthew-Maich, L. Harris, J. Ploeg, M. Markle-Reid, R. Valaitis, S. Ibrahim, A. Gafni, and S. Isaacs, "Designing, implementing, and evaluating mobile health technologies for managing chronic conditions in older adults: A scoping review," *JMIR mHealth and uHealth*, vol. 4, no. 2, pp. 1–18, 2016.

[8] J. Joe and G. Demiris, "Older adults and mobile phones for health: A review," *Journal of Biomedical Informatics*, vol. 46, no. 5, pp. 947–954, 2013.

[9] L. Pfaeffli Dale, R. Dobson, R. Whittaker, and R. Maddison, "The effectiveness of mobile-health behaviour change interventions for cardiovascular disease self-management: A systematic review," *European Journal of Preventive Cardiology*, vol. 23, no. 8, pp. 801–817, 2016.

[10] B. E. Dicianno, B. Parmanto, A. D. Fairman, T. M. Crytzer, D. X. Yu, G. Pramana, D. Coughenour, and A. A. Petrazzi, "Perspectives on the evolution of mobile (mHealth) technologies and application to rehabilitation," *Physical Therapy*, vol. 95, no. 3, pp. 397–405, 2015.

[11] D. M. Zulman, E. C. Jenchura, D. M. Cohen, E. T. Lewis, T. K. Houston, and S. M. Asch, "How Can eHealth Technology Address Challenges Related to Multimorbidity? Perspectives from Patients with Multiple Chronic Conditions," *Journal of General Internal Medicine*, vol. 30, no. 8, pp. 1063–1070, 2015.

[12] S. Kumar, W. J. Nilsen, A. Abernethy, A. Atienza, K. Patrick, M. Pavel, W. T. Riley, A. Shar, B. Spring, D. Spruijt-Metz, D. Hedeker, V. Honavar, R. Kravitz, R. Craig Lefebvre, D. C. Mohr, S. A. Murphy, C. Quinn, V. Shusterman, and D. Swendeman, "Mobile health technology evaluation: The mhealth evidence workshop," *American Journal of Preventive Medicine*, vol. 45, no. 2, pp. 228–236, 2013.

[13] G. S. Karanasiou, E. E. Tripoliti, Y. Goletsis, F. G. Kalatzis, K. K. Naka, R. Fuoco, W. Miekisch, K. E. Tzannetis, S. G. Zervos, J. R. Bausells, A. Errachid, and D. I. Fotiadis, "HEARTEN: An integrated mHealth platform for holistic HF management," *2018 IEEE EMBS International Conference on Biomedical and Health Informatics, BHI 2018*, pp. 295–298, 2018.

[14] WHO Global Observatory for eHealth, "mHealth: new horizons for health through mobile technologies: second global survey on eHealth," 2011.

[15] Y. Fukazawa, N. Yamamoto, T. Hamatani, K. Ochiai, A. Uchiyama, and K. Ohta, "Smartphone-based mental state estimation: A survey from a machine learning perspective," *Journal of Information Processing*, vol. 28, pp. 16–30, 2020.

[16] N. Bidargaddi, D. Almirall, S. Murphy, I. Nahum-Shani, M. Kovalcik, T. Pituch, H. Maaieh, and V. Strecher, "To prompt or not to prompt? A microrandomized trial of time-varying push notifications to increase proximal engagement with a mobile health app," *JMIR mHealth and uHealth*, vol. 6, no. 11, 2018.

[17] S. R. Toukhsati, A. Driscoll, and D. L. Hare, "Patient self-management in chronic heart failure–establishing concordance between guidelines and practice," *Cardiac Failure Review*, vol. 1, no. 2, pp. 128–131, 2015.

[18] K. Wang, D. S. Varma, and M. Prosperi, "A systematic review of the effectiveness of mobile apps for monitoring and management of mental health symptoms or disorders," *Journal of Psychiatric Research*, vol. 107, pp. 73–78, 2018.

[19] ITU, "Digital Trends in Europe 2021 - ICT trends and developments in Europe, 2017-2020," 2021.

[20] Market Data Forecast, “Global mHealth market / Mobile Health Market Size, Share, Trends, Growth & COVID-19 Impact Analysis Report - Segmented By Type, Services, Application & Region – Industry Forecast (2021 to 2026),” 2021.

[21] A. Aguilera, C. A. Figueroa, R. Hernandez-Ramos, U. Sarkar, A. Cemballi, L. Gomez-Pathak, J. Miramontes, E. Yom-Tov, B. Chakraborty, X. Yan, J. Xu, A. Modiri, J. Aggarwal, J. Jay Williams, and C. R. Lyles, “MHealth app using machine learning to increase physical activity in diabetes and depression: Clinical trial protocol for the DIAMANTE Study,” *BMJ Open*, vol. 10, no. 8, pp. 1–9, 2020.

[22] J. Seppälä, I. De Vita, T. Jämsä, J. Miettunen, M. Isohanni, K. Rubinstein, Y. Feldman, E. Grasa, I. Corripio, J. Berdun, E. D’Amico, M. R. Group, and M. Bulgheroni, “Mobile phone and wearable sensor-based mHealth approach for psychiatric disorders and symptoms: Systematic review,” *Journal of Medical Internet Research*, vol. 6, no. 2, pp. 1–14, 2019.

[23] M. N. Burns, M. Begale, J. Duffecy, D. Gergle, C. J. Karr, E. Giangrande, and D. C. Mohr, “Harnessing Context Sensing to Develop a Mobile Intervention for Depression,” *Journal of Medical Internet Research*, vol. 13, no. 3, 2011.

[24] H. Thórarinsdóttir, L. V. Kessing, and M. Faurholt-Jepsen, “Smartphone-based self-assessment of stress in healthy adult individuals: a systematic review,” *Journal of Medical Internet Research*, vol. 19, no. 2, pp. 1–13, 2017.

[25] Markets and Markets, “mHealth Solutions Market by Connected Devices (Glucose & Blood Pressure Monitor, Peak Flow Meter, Pulse Oximeter), Apps (Weight Loss, Women Health, Diabetes Management, Mental Health), Services (Remote Monitoring, Consultation) - Global Forecast to 2025,” 2020.

[26] S. Vishwanath, K. Vaidya, R. Nawal, S. Parthasarathy, and S. Verma, “Touching lives through mobile health-Assessment of the global market opportunity,” *PwC: Bangalore (India)*, 2012.

[27] N. Bidargaddi, G. Schrader, P. Klasnja, J. Licinio, and S. Murphy, “Designing m-Health interventions for precision mental health support,” *Translational Psychiatry*, vol. 10, no. 1, pp. 1–8, 2020.

[28] S. Berrouiguet, M. L. Barrigón, J. L. Castroman, P. Courtet, A. Artés-Rodríguez, and E. Baca-García, “Combining mobile-health (mHealth) and artificial intelligence (AI) methods to avoid suicide attempts: The Smartcrises study protocol,” *BMC Psychiatry*, vol. 19, no. 1, pp. 1–9, 2019.

[29] M. Fiordelli, N. Diviani, and P. J. Schulz, “Mapping mhealth research: A decade of evolution,” *Journal of Medical Internet Research*, vol. 15, no. 5, pp. 1–14, 2013.

[30] A. K. Triantafyllidis and A. Tsanas, “Applications of machine learning in real-life digital health interventions: Review of the literature,” *Journal of Medical Internet Research*, vol. 21, no. 4, pp. 1–9, 2019.

[31] A. Mehrotra, M. Musolesi, R. Hendley, and V. Pejovic, “Designing content-driven intelligent notification mechanisms for mobile applications,” *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 813–824, 2015.

[32] W. Van Woensel, P. C. Roy, S. R. Abidi, and S. S. Abidi, “A Mobile and Intelligent Patient Diary for Chronic Disease Self-Management,” in *MedInfo*, 2015, pp. 118–122.

[33] V. Pejovic and M. Musolesi, “InterruptMe: Designing intelligent prompting mechanisms for pervasive applications,” *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 897–908, 2014.

[34] M. Pielot, T. Dingler, J. S. Pedro, and N. Oliver, “When attention is not scarce-detecting boredom from mobile phone usage,” *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 825–836, 2015.

[35] J. Ho and S. S. Intille, “Using context-aware computing to reduce the perceived burden of interruptions from mobile devices,” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 909–918, 2005.

[36] B. J. Ho, B. Balaji, M. Koseoglu, and M. Srivastava, “Nurture: Notifying users at the right time using reinforcement learning,” *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pp. 1194–1201, 2018.

[37] S. Kumar, W. Nilsen, M. Pavel, and M. Srivastava, “Mobile health: Revolutionizing healthcare through transdisciplinary research,” *Computer*, vol. 46, no. 1, pp. 28–35, 2012.

[38] A. S. Shirazi, N. Henze, T. Dingler, M. Pielot, D. Weber, and A. Schmidt, “Large-scale assessment of mobile notifications,” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3055–3064, 2014.

[39] M. Pielot, K. Church, and R. De Oliveira, “An in-situ study of mobile phone notifications,” *Proceedings of the 16th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services*, pp. 233–242, 2014.

[40] L. G. Morrison, C. Hargood, V. Pejovic, A. W. Geraghty, S. Lloyd, N. Goodman, D. T. Michaelides, A. Weston, M. Musolesi, M. J. Weal, and L. Yardley, “The effect of timing and frequency of push

notifications on usage of a smartphone-based stress management intervention: An exploratory trial," *PLoS ONE*, vol. 12, no. 1, pp. 1–15, 2017.

[41] A. Mehrotra, V. Pejovic, J. Vermeulen, R. Hendley, and M. Musolesi, "My phone and me: Understanding people's receptivity to mobile notifications," *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 1021–1032, 2016.

[42] J. E. Fischer, N. Yee, V. Bellotti, N. Good, S. Benford, and C. Greenhalgh, "Effects of content and time of delivery on receptivity to mobile interruptions," *Proceedings of the 12th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pp. 103–112, 2010.

[43] M. M. Theilig, J. J. Korbel, G. Mayer, C. Hoffmann, and R. Zarnekow, "Employing Environmental Data and Machine Learning to Improve Mobile Health Receptivity," *IEEE Access*, vol. 7, pp. 179 823–179 841, 2019.

[44] S. T. Iqbal and E. Horvitz, "Notifications and awareness: A field study of alert usage and preferences," *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, pp. 27–30, 2010.

[45] B. P. Bailey and J. 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*, vol. 22, no. 4, pp. 685–708, 2006.

[46] E. Cutrell, M. Czerwinski, and E. Horvitz, "Notification, Disruption, and Memory: Effects of Messaging Interruptions on Memory and Performance," *Human-Computer Interaction: Interact 2001*, vol. 1, pp. 263–269, 2001.

[47] B. P. Bailey, J. A. Konstan, and J. V. Carlis, "Measuring the effects of interruptions on task performance in the user interface," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 757–762, 2000.

[48] J. E. Fischer, C. Greenhalgh, and S. Benford, "Investigating episodes of mobile phone activity as indicators of opportune moments to deliver notifications," *Proceedings of the 13th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pp. 181–190, 2011.

[49] A. Mehrotra and M. Musolesi, "Intelligent Notification Systems: A Survey of the State of the Art and Research Challenges," vol. 1, no. 1, pp. 1–26, 2017.

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

[51] T. Okoshi, J. Ramos, H. Nozaki, J. Nakazawa, and A. K. Dey, “Reducing users’ perceived mental effort due to interruptive notifications in multi-device mobile environments,” *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pp. 475–486, 2015.

[52] S. G. Hart and L. E. Staveland, “Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research,” *Advances in Psychology*, vol. 52, pp. 139–183, 1988.

[53] A. Holzinger, “Interactive machine learning for health informatics: when do we need the human-in-the-loop?” *Brain Informatics*, vol. 3, no. 2, pp. 119–131, 2016.

[54] T. Rachad and A. Idri, “Intelligent Mobile Applications: A Systematic Mapping Study,” *Mobile Information Systems*, vol. 2020, 2020.

[55] K. C. Tsang, H. Pinnock, A. M. Wilson, and S. Ahmar Shah, “Application of Machine Learning to Support Self-Management of Asthma with mHealth,” *Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, pp. 5673–5677, 2020.

[56] R. Manuvinakurike, W. F. Velicer, and T. W. Bickmore, “Automated indexing of internet stories for health behavior change: weight loss attitude pilot study,” *Journal of Medical Internet Research*, vol. 16, no. 12, pp. 1–12, 2014.

[57] A. A. Ali, S. M. Hossain, K. Hovsepian, M. M. Rahman, K. Plarre, and S. Kumar, “mPuff: Automated detection of cigarette smoking puffs from respiration measurements,” *Proceedings of the 11th International Conference on Information Processing in Sensor Networks*, pp. 269–280, 2012.

[58] U. Reimer, E. Maier, and T. Ulmer, “SmartCoping: A Mobile Solution for Recognizing Stress and Coping with It,” in *Delivering Superior Health and Wellness Management with IoT and Analytics*. Springer, 2020, pp. 119–143.

[59] K. Plarre, A. Raij, S. M. Hossain, A. A. Ali, M. Nakajima, M. Al’Absi, E. Ertin, T. Kamarck, S. Kumar, M. Scott, D. Siewiorek, A. Smailagic, and L. E. Wittmers, “Continuous inference of psychological stress from sensory measurements collected in the natural environment,” *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks, IPSN’11*, pp. 97–108, 2011.

[60] G. A. O’Reilly and D. Spruijt-Metz, “Current mHealth technologies for physical activity assessment and promotion,” *American Journal of Preventive Medicine*, vol. 45, no. 4, pp. 501–507, 2013.

[61] R. S. Istepanian and T. Al-Anzi, “m-Health 2.0: New perspectives on mobile health, machine learning and big data analytics,” *Methods*, vol. 151, pp. 34–40, 2018.

[62] S. M. Noar, C. N. Benac, and M. S. Harris, “Does Tailoring Matter? Meta-Analytic Review of Tailored Print Health Behavior Change Interventions,” *Psychological Bulletin*, vol. 133, no. 4, pp. 673–693, 2007.

[63] C. R. Lyles, U. Sarkar, and C. Y. Osborn, “Getting a Technology-Based Diabetes Intervention Ready for Prime Time: a Review of Usability Testing Studies,” *Current Diabetes Reports*, vol. 14, no. 10, pp. 1–21, 2014.

[64] S. N. Smith, A. J. Lee, K. Hall, N. J. Seewald, A. Boruvka, S. A. Murphy, and P. Klasnja, “Design Lessons from a Micro-Randomized Pilot Study in Mobile Health,” in *Mobile Health*. Springer, 2017, pp. 59–82.

[65] A. G. B. Richard S. Sutton, *Reinforcement learning : an introduction*, 2nd ed. MIT Press, 2018.

[66] Mark Humphrys, “Action Selection methods using Reinforcement Learning,” *From Animals to Animats*, vol. 4, pp. 135–144, 1996.

[67] F. Cruz, P. Wuppen, A. Fazrie, C. Weber, and S. Wermter, “Action Selection Methods in a Robotic Reinforcement Learning Scenario,” in *2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, 2018, pp. 1–6.

[68] C. J. C. H. Watkins, “Learning from delayed rewards,” PhD thesis, University of Cambridge, 1989.

[69] P. Watkins, Christopher JCH and Dayan, “Q-learning,” *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[70] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[71] L.-J. Lin, “Self-improvement Based On Reinforcement Learning, Planning and Teaching,” *Machine Learning*, vol. 8, pp. 293–321, 1992.

[72] L.-j. Lin, “Reinforcement Learning for Robots Using Neural Networks,” Ph.D. dissertation, 1993.

[73] “Random - generate pseudo-random numbers.” [Online]. Available: https://docs.python.org/3/library/random.html

[74] M. Matsumoto and T. Nishimura, “Mersenne Twister : A 623-dimensionally equidistributed uniform pseudorandom number generator,” *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 8, no. 1, pp. 3–30, 1998.

[75] “Numpy.” [Online]. Available: https://numpy.org/

[76] “Math - mathematical functions.” [Online]. Available: https://docs.python.org/3/library/math.html

[77] “Pandas.” [Online]. Available: https://pandas.pydata.org/

[78] “Tensorflow.” [Online]. Available: https://www.tensorflow.org/

[79] “Gym - a toolkit for developing and comparing reinforcement learning algorithms.” [Online]. Available: https://gym.openai.com/

[80] D. C. McFarlane and K. A. Latorella, “The Scope and Importance of Human Interruption in Human-Computer Interaction Design,” *Human-Computer Interaction*, vol. 17, no. 1, pp. 1–61, 2002.

Appendix A

Contents

A.1 ExtraSensory Dataset Labels

Number	Label	Label Type	$P(A \mid L)$
1	OR_indoors	Location	0.5
2	LOC_home	Location	0.6
3	SITTING	Primary Activity	0.6
4	PHONE_ON_TABLE	Secondary Activity	0.7
5	LYING_DOWN	Primary Activity	0.6
6	SLEEPING	Secondary Activity	0.01
7	AT_SCHOOL	Location	0.5
8	COMPUTER_WORK	Secondary Activity	0.3
9	OR_standing	Primary Activity	0.5
10	TALKING	Secondary Activity	0.3
11	LOC_main_workplace	Location	0.5
12	WITH_FRIENDS	Secondary Activity	0.3
13	PHONE_IN_POCKET	Secondary Activity	0.4
14	FIX_walking	Primary Activity	0.5
15	SURFING_THE_INTERNET	Secondary Activity	0.8
16	EATING	Secondary Activity	0.8
17	PHONE_IN_HAND	Secondary Activity	0.8
18	WATCHING_TV	Secondary Activity	0.9
19	OR_outside	Location	0.5
20	PHONE_IN_BAG	Secondary Activity	0.4
21	OR_exercise	Primary Activity	0.15
22	DRIVE_-_I_M_THE_DRIVER	Secondary Activity	0.1
23	WITH_CO-WORKERS	Secondary Activity	0.1
24	IN_CLASS	Location	0.2
25	IN_A_CAR	Location	0.5
26	IN_A_MEETING	Location	0.08
27	BICYCLING	Primary Activity	0.05
28	COOKING	Secondary Activity	0.7
29	LAB_WORK	Secondary Activity	0.1
30	CLEANING	Secondary Activity	0.2
31	GROOMING	Secondary Activity	0.18
32	TOILET	Secondary Activity	0.9
33	DRIVE_-_I_M_A_PASSENGER	Secondary Activity	0.6
34	DRESSING	Secondary Activity	0.2
35	FIX_restaurant	Location	0.6
36	BATHING_-_SHOWER	Secondary Activity	0.1
37	SHOPPING	Secondary Activity	0.3
38	ON_A_BUS	Location	0.7
39	AT_A_PARTY	Location	0.3
40	DRINKING__ALCOHOL_	Secondary Activity	0.6
41	WASHING_DISHES	Secondary Activity	0.06
42	AT_THE_GYM	Location	0.4
43	FIX_running	Primary Activity	0.1
44	STROLLING	Primary Activity	0.8
45	STAIRS_-_GOING_UP	Secondary Activity	0.5
46	STAIRS_-_GOING_DOWN	Secondary Activity	0.5
47	SINGING	Secondary Activity	0.9

Continued on next page

Table A.1: Table of labels used in the ExtraSensory Dataset [1].

Table A.1 – continued from previous page

Number	Label	Label Type	$P(A \mid L)$
48	LOC_beach	Location	0.8
49	DOING_LAUNDRY	Secondary Activity	0.8
50	AT_A_BAR	Location	0.4
51	ELEVATOR	Location	0.8

Table A.1: Table of labels used in the ExtraSensory Dataset [1].

A.2 User Characterization

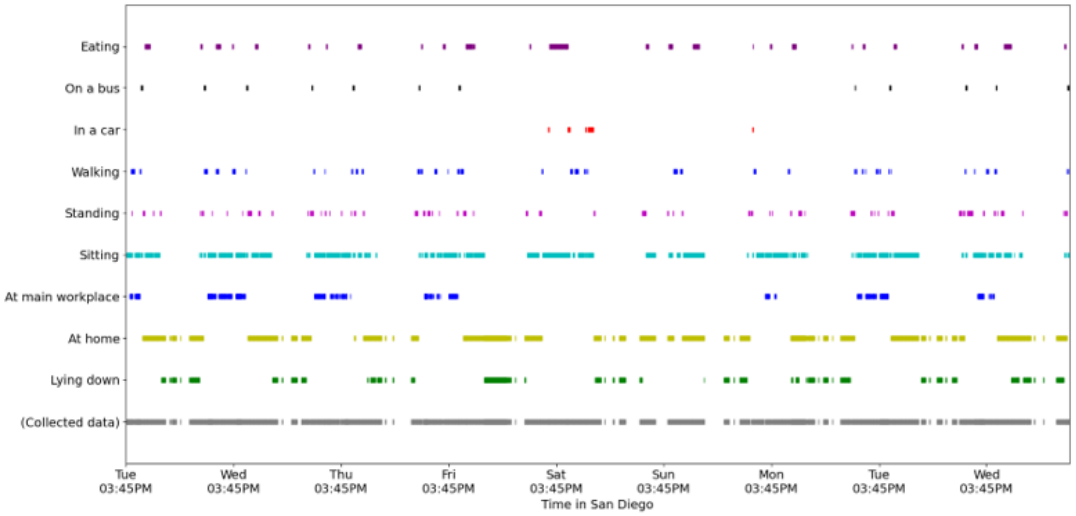
User \Primary Activity (hours)	Lying Down	Walking or Running	Standing	Sitting	Total number of hours of recorded data in 7 days
User 1	46.8048	9.9456	14.448	96.5496	167.748
User 2	40.9752	22.1424	22.3272	78.3888	163.8336
User 3	46.0656	4.704	19.32	75.6672	145.7568

Table A.2: User Characterization - Main Primary Activities.

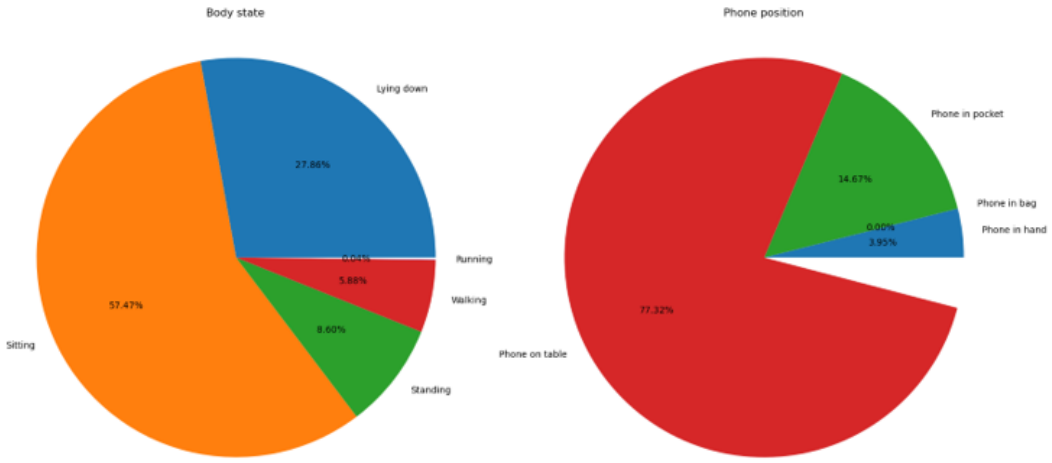
User \Main Locations (hours)	At Home	At the Workplace	In a car	On a bus	Total number of hours of recorded data in 7 days
User 1	89.9808	36.9768	1.8816	2.6208	131.46
User 2	110.292	0	0.8232	0.7224	111.8376
User 3	3.4	39.9	0	0.672	43.972

Table A.3: User Characterization - Main Locations.

In the following images (Figures A.1 to A.3), the users' lifestyle can be seen in further detail:

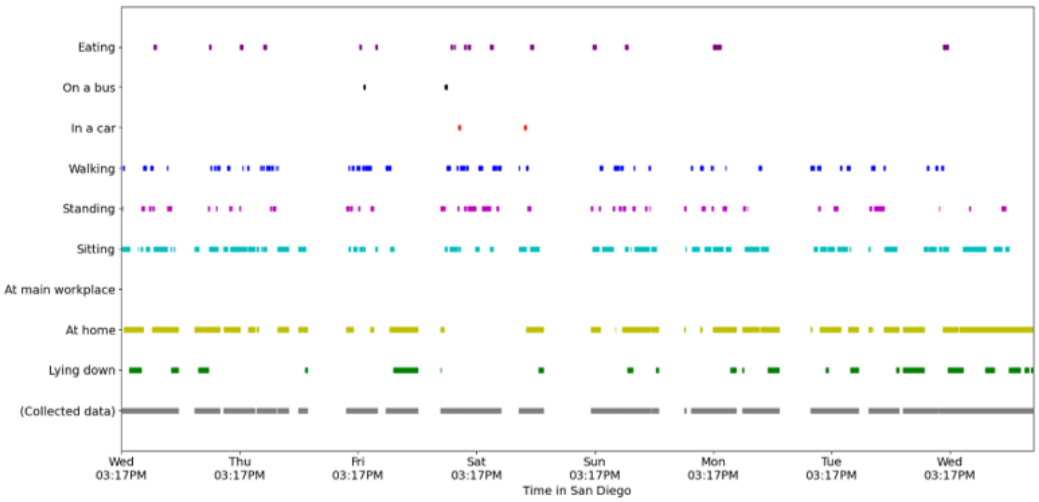


(a) Main Activities

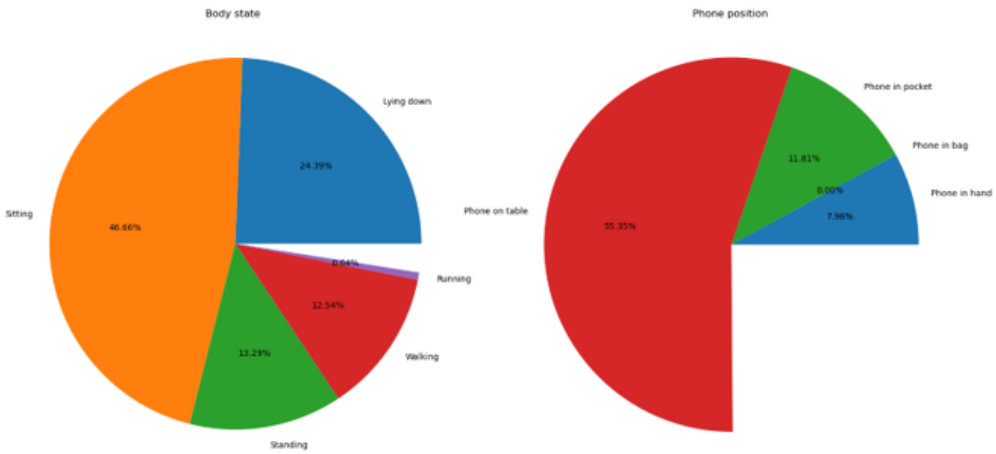


(b) Phone State and Position

Figure A.1: User 1 Characterization.

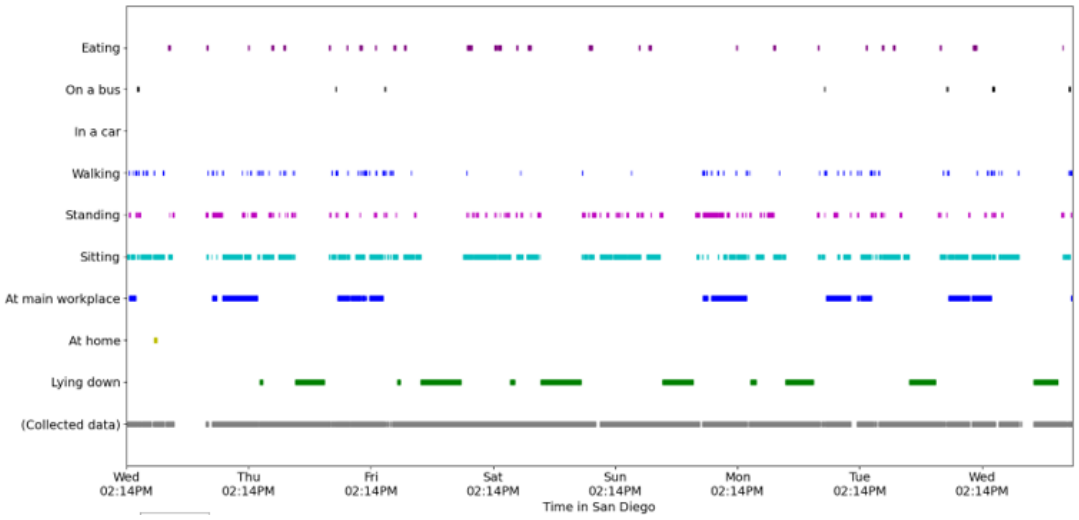


(a) Main Activities

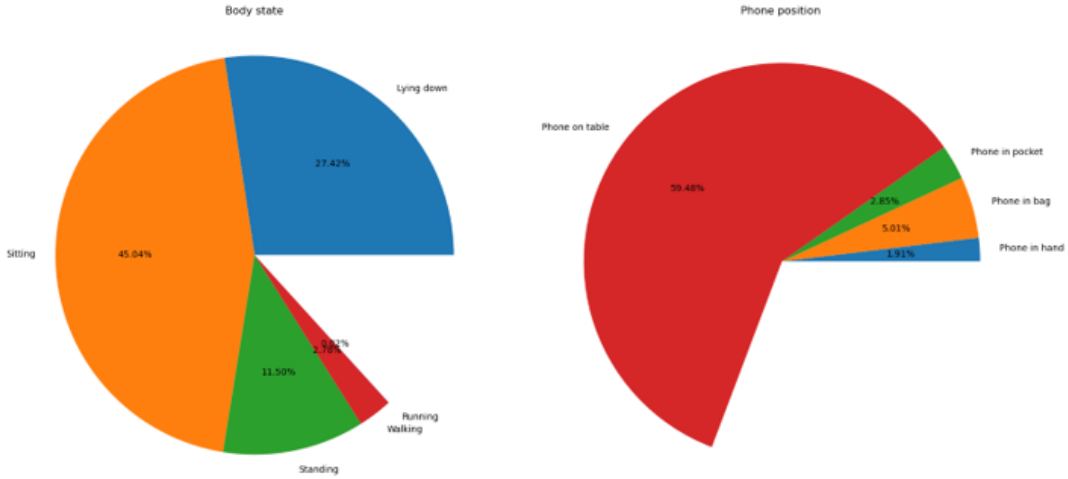


(b) Phone State and Position

Figure A.2: User 2 Characterization.



(a) Main Activities



(b) Phone State and Position

Figure A.3: User 3 Characterization.