Active Data Collection of Health Data in Mobile Devices

Ana Rita Bamansá Siles Machado[1], Heitor Cardoso[2], Plinio Moreno[2] and Alexandre Bernardino[2]

[1]*Instituto Superior Técnico, Universidade de Lisboa, 1049-001 Lisboa, Portugal*

[2]*Institute for Systems and Robotics, Instituto Superior Técnico, Universidade de Lisboa, Torre Norte Piso 7, 1049-001 Lisboa, Portugal*

Keywords: mHealth, Notifications, Machine Learning, Personalization, Reinforcement Learning, Receptivity.

Abstract: This paper aims to develop an intelligent notification system to help sustain user engagement in mHealth applications, specifically those that support self-management. We rely on Reinforcement Learning (RL), an approach where agent learns by exploration the most opportune time to perform a questionnaire, throughout their day, only from easily obtainable non-sensitive data and usage history. This history allows the agent to remember how the user reacts or has reacted in the past to its actions. We consider several options on algorithm, state representation and reward function under the RL umbrella (Upper Confidence Bound, Tabular Q-learning and Deep Q-learning). In addition, a simulator was developed to mimic the behavior of a typical user and utilized to test all possible combinations with users experiencing distinct lifestyles. We obtain promising promising results, which still requiring further testing to be fully validated. We demonstrate that an efficient and well-balanced notification system can be built with simple formulations of an RL problem and algorithm. Furthermore, our approach does not require to have access to sensitive user data. 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.

1 INTRODUCTION

Mobile health, or mHealth, is defined as "medical and public health practice supported by mobile devices" such as phones, wearables or other patient monitoring devices (WHO Global Observatory for eHealth, 2011) and is a great vehicle for the support of self-management in Noncommunicable diseases (NCDs).

NCDs, also known as chronic conditions (Fukazawa et al., 2020), such as cancer, diabetes, stroke, and other chronic respiratory or cardiovascular diseases, are the leading causes of death and disability worldwide. 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 (Geneva: WHO, 2020). The ability for patients to employ self-management is now more vital than ever and many studies have already shown promise for its application in helping manage these chronic conditions (Cornet and Holden, 2018). However, 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. Impact of such self-management approaches requires widespread user adoption and engagement (Vishwanath et al., 2012).

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 (Fiordelli et al., 2013). Nonetheless, the risk of intrusiveness into daily life is imminent. Furthermore, consumers are known to highly dislike excessive or inopportune notifications, primarily when originated by machines (Mehrotra et al., 2015). For these reasons, mHealth applications must function and communicate without burdening the consumer. Hence, this paper focuses on developing an intelligent notification system that intends to increase continued engagement by helping applications communicate with users when they are receptive, not bothering them on inconvenient occasions. The purpose of this paper is to develop a mechanism that is able to identify opportune moments for notifications. This goal entails challenging cross-disciplinary subjects such as information technology, medicine, and psychology,

Machado, A., Cardoso, H., Moreno, P. and Bernardino, A.
Active Data Collection of Health Data in Mobile Devices.
DOI: 10.5220/0011300700003277
In *Proceedings of the 3rd International Conference on Deep Learning Theory and Applications (DeLTA 2022)*, pages 160-167
ISBN: 978-989-758-584-5; ISSN: 2184-9277

which plays a big part in understanding human behavior. Throughout an average day, users can get over 50 notifications on their phones. Hence, feeling overwhelmed or experiencing growing negative sentiments towards individual notifications or apps is expected (Mehrotra et al., 2015). Receptivity is defined as the degree to which a user considers that a notification is received at an opportune time. 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 (Mehrotra et al., 2015), such as location, movement, time or psychological state.

Most of current applications employ a basic interaction model, which assumes the availability of the user at any time for engagement with the device. The potentially disruptive impact of these applications should be compensated with the customization of notifications' characteristics (such as presentation or alert type) (Mehrotra et al., 2016), content (Mehrotra et al., 2015), and an intelligent approach to deliver them (Mehrotra and Musolesi, 2017). Therefore, systems that attempt to handle such notifications intelligently are increasingly relevant. Although many studies have been published on this kind of system, most of the existing prior work can be divided into (Ho et al., 2018): detecting transitions between activities, assuming these represent the most opportune timings in a user's routine, or inferring receptivity from the user's context.

Systems such as these have resorted primarily to machine learning (ML) techniques due to their capability of discovering patterns in data (Mehrotra et al., 2015). ML techniques can be divided into five: Supervised (SL), Unsupervised, Semi-Supervised, Active and Reinforcement learning (RL). SL is the most common approach, presumably due to the tendency of seeing the problem at hand as the need to classify users, their preferences, or even labelling moments as opportune or inconvenient. Albeit its known capacity for swift adaptation to dynamic, complex environments, RL is more complicated to apply than SL, leaving it with few implementation attempts in the mHealth field (Rachad and Idri, 2020).

2 METHODOLOGY

Through the application of RL, our system aims to learn user preferences, routines, and habits merely from notification interaction data. 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. The RL agent considers an answered notification as the 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 is at the end of the day (24h). Tasks such as this are called episodic tasks (Richard S. Sutton, 2018).

The agent's main objective is to decide whether to send a notification or stay silent, by observing the user and the environment's state. 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 that penalize the agent. The agent's behavior changes accordingly, always intending to increase the long-run sum of rewards (reinforcement signals).

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.

2.1 RL Algorithms

In order to perform experiments that are as varied as possible, several consensually recommended algorithms were implemented.

Upper Confidence Bound (UCB) - UCB emerges as a widely accepted nonassociative bandit[1], as it considers the problem as only a single state. UCB achieves exploration by subtly favoring the selection of actions that have the potential to be optimal and have been employed the less (Richard S. Sutton, 2018). To do so, it applies the selection rule:

$$A_t \doteq \underset{a}{argmax} \left[Q_t(a) + c\sqrt{\frac{\ln(t)}{N_t(a)}} \right] \tag{1}$$

where A and a represent actions, t represents the current timestep, Q is the action-value function and c, the confidence level that controls the degree of exploration. Finally, $N_t(a)$ represents the number of times action a has been selected prior to time t.

In (1), 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. Given a large enough time, UCB executes all

[1]Bandit problems are RL problems where the agent learns to act in a single state setting, not requiring an association between actions and states (nonassociative).

the available actions, guaranteeing that the agent explores the action space properly. As time goes on and different actions are performed, to each, the sum of received rewards and the number of selections are associated. With these values, the action-value function Q is updated at each timestep.

Since UCB is a single state algorithm, it learns what is better suited for that state only, which in this paper's context would be very restrictive. To mitigate this, we consider several instances of UCB running at every possible states, on two versions. On the UCB Day version, a different UCB instance is applied to each hour of the day. Thus, the agent learns 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 instance is applied to each hour of each day, during a week. Considering that the week has 7 days, and 24 instances 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) - Q-learning was initially defined by (Watkins,, 1992) as follows:

$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha[R_{t+1} + \gamma \max_a(Q(S_{t+1},a)) - Q(S_t,A_t)], \tag{2}$$

where A and a represent actions, S represents the state and R the reward. Additionally, t represents the current timestep, Q is the action-value function for each state-action pair, α is the learning rate and, lastly, γ is the discount factor. The learning rate, α, determines when Q-values are updated, overriding older 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. In (2), the policy is greedy because 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. However, different policies can be applied to actually choose the desired action. This choice should take into consideration the context and particularities of the problem in question. In our work, the ε-greedy policy, where the agent behaves mostly in a greedy way but occasionally, and with a small probability ($\varepsilon > 0$), selects a random action, showed promising results and was henceforth applied.

Deep Q-Learning (DQN) with Experience Replay - Developed by (Mnih et al., 2015), the Deep Q-learning agent, 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. 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:

$$Q_{target}(t) = \begin{cases} r_t, \\ \quad \text{for terminal } \phi_{t+1} \\ r_t + \gamma \max_{a'}(Q_\theta(\phi_{t+1}, a')), \\ \quad \text{for non-terminal } \phi_{t+1} \end{cases} \tag{3a}$$

$$Loss(\theta) = \sum (Q_{target}(t) - Q_\theta(\phi_t, a_t))^2 \tag{3b}$$

where a represents an action, ϕ represents the state and R the reward. Additionally, t represents the current timestep, Q is the action-value function for each state-action pair, and θ represent the network weights.

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 (Lin, 1993), where the agent memorizes the state, action, and effect of that same action in the environment for every timestep. After completing an episode, it replays the gathered experiences 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.

2.2 State Representation

We consider four representations: S1 and S3 have similar formats, both containing the time of the day in minutes, the number of notifications already sent and answered that day, and the last user reaction. The difference between these states is that S1 also contains the day of the week, depicted by values from 0 to 6, allowing for a representation of a weekly routine instead of a simple daily routine such as S1 permits. S2 and S4 were born from a similar approach, both containing an array of 24 elements where all positions start as 0 and then, throughout the day, each element may change depending on the outcome of the action chosen at every hour (below, the numbers used to express each user reaction are described). S2 has an additional element which, again and with the same aim as before, represents the day of the week.

The developed states resort to easily obtainable information, 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. With the purpose of recording users' reactions to the actions of the agent, 3 options were defined and associated with a value: 0, meaning that in the last timestep, a notification was sent and ignored or dismissed; 1,a notification was sent and positively addressed ; 3, a notification was not sent.

2.3 Reward Definition

The various types rewards of our experiments are structured in the following manner: when a notification is sent and the user does not answer, the agent receives reward a. However, if the user responds then the received reward value is b. Contrarily, if a notification is not sent the agent receives c. Lastly, if the episode, in this context a day, ends without having achieved the goal of one answer then d is received. Thus, the rewards assume values in the set $R = \{a,b,c,d\}$. We define the following alternatives for the values of $\{a,b,c,d\}$: R1 = $\{-1,2,-1,-2\}$; R3 = $\{-2,2,-1,-3\}$; R5 = $\{-2,2,0,-3\}$; R6 = $\{-3,2,0,-3\}$.

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 by sending notifications that go unanswered.

2.4 Environment Model

We assume that no difference exists between ignoring a message or explicitly dismissing it, considering both as "No Answer". In this initial approach, 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.

2.4.1 Behavior Model

This model reflects a users' routine, for example, the activities performed, their duration, and the user's location. It mirrors the ExtraSensory dataset (Vaizman et al., 2017), which aggregates daily traces of 60 participants. Measurements from smartphones and smartwatches were collected, along with self-reported labels. Since this data was collected in the wild, its reliability is not perfect; after processed and cleaned, it considers 51 possible tags, shown in Appendix A (15 locations, 8 primary activities, 28 secondary ones). These include primary activities, which describe movement or posture and are mutually exclusive, and secondary activities, which 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 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. 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.

2.4.2 Response Model

The response model simulates how a user responds to a notification in any given context, originating the observations that our agent receives. In the literature, a set of behaviors that researchers consensually agree users tend to show were considered when implementing this model (Mehrotra et al., 2015; Mehrotra et al., 2016; Mehrotra and Musolesi, 2017).

Firstly, when the behavior model presents labels such as sleeping, which ensure an inability to answer, the simulator does not respond to notifications. In the case of 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. Secondly, 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. Thirdly, a component (β), defined as the exponential decay in (4), is used to convey the diminishing desire to use the app that most users would experience as the number of daily notifications rises.

$$\beta(n_t) = P(Answer \mid n_t) = e^{-\lambda n_t} \tag{4}$$

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.

Each user has a predefined prior probability of answering $P(A)$ and not answering $P(\overline{A})$. This value represents a person's predisposition to be on their phone and regularly use a mHealth application. We assume a fixed value for each simulated user.

Assuming statistical independence between labels and following the Naive Bayes probability model (5), the probability of the user answering or not is as follows:

$$\begin{aligned}
&P(C \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \\
&\propto P(C, L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \\
&\propto P(C)P(L_0 \mid C)...P(L_i \mid C)P(\overline{L_{i+1}} \mid C)...P(\overline{L_{l_t}} \mid C) \\
&\propto P(C) \prod_{j=0}^{i} \left[P(L_j \mid C)\right] \prod_{k=i+1}^{l_t} \left[P(\overline{L_k} \mid C)\right]
\end{aligned} \tag{5}$$

A set of L_t labels, provided by the behavior model represents this context. 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 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 presented in (5).

The values of $P(L \mid C)$ and $P(\overline{L} \mid C)$ are unknown. To compute them, we use the Bayes' theorem, $P(L \mid C) = P(C \mid L)P(L)/P(C)$. Considering the conditional probability formula, $P(C \mid L) = P(C, L)/P(L)$, and $P(C, L) + P(C, \overline{L}) = P(C)$, we compute $P(\overline{L} \mid C)$ using only $P(L \mid C)$. 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.

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 ($P(\overline{A} \mid L)$).

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 (6).

$$P(L_k) = \frac{N_{L_k}}{N_L} \tag{6}$$

The result is user-dependent 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.

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 unknown values of $P(L \mid C)$ and $P(\overline{L} \mid C)$ are easily obtained. We add the parameter β that represents user discontentment with notification volume, in (7) and (8), finalizing our expressions as

$$\begin{aligned}
&P(A \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \propto \\
&\beta \left[P(A) \prod_{j=0}^{i} \left[P(L_j \mid A)\right] * \prod_{k=i+1}^{l_t} \left[P(\overline{L_k} \mid A)\right] \right]
\end{aligned} \tag{7}$$

$$\begin{aligned}
&P(\overline{A} \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \propto \\
&(1-\beta) \left[P(\overline{A}) \prod_{j=0}^{i} \left[P(L_j \mid \overline{A})\right] * \prod_{k=i+1}^{l_t} \left[P(\overline{L_k} \mid \overline{A})\right] \right]
\end{aligned} \tag{8}$$

For each instance, the above presented factors are estimated, normalized, and, resorting to a simple sampling method, the simulator's response is determined.

$$\begin{aligned}
&X \sim U(0,1) \\
&\hat{c} \in \{A, \overline{A}\} \sim \underset{C \in \{A, \overline{A}\}}{\left\| P(C \mid L_0, ..., L_i, \overline{L_{i+1}}, ..., \overline{L_{l_t}}) \right\|} \leqslant X
\end{aligned} \tag{9}$$

We believe this sampling technique allows us to reflect the ambiguity of users more accurately. $\hat{c}$ represents the class that defines the simulator's response. Depending on the simulator's reaction and the state of the environment, the algorithm then obtains the respective reward and adjusts the strategy accordingly.

3 EXPERIMENTS

3.1 Model Initialization Methods

One of the main objectives of this paper is to analyze the efficiency levels that algorithms can obtain when models are initialized in different manners. **No Previous Knowledge Models (Online Learning) -** The models start with no prior knowledge, learning only from interaction with a specific user. We expect that this model adapts better to the user, taking longer to reach the better customization. **Previously trained models (Offline Learning) -** Here, 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. This approach should reach acceptable results right away. However, the method will not adapt to user input. **Previously trained adaptive models (Combination of Offline and Online Learning) -** In this case, models are likewise trained 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.

3.2 Daily vs. Weekly Routine

By applying the different state representations of Q-learning and DQN and the different formulations of

UCB, 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 instead of a typical day. 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.

3.3 Performance Metrics

As performance metrics of our algorithms, we selected two: Goal Achievement Rate (G_r) and Notification Volume (N_v).

$$G_r = \frac{N_A}{N_{Days}} \tag{10}$$

$$N_v = \frac{N_{Sent}}{N_{Days}} = \frac{\sum_{i=0}^{Days}(N_{A_i} + N_{\overline{A}_i})}{N_{Days}} \tag{11}$$

G_r, in (10), is the fraction of accepted notifications (N_A) over the number of episodes being tested (N_{Days}, each episode representing a day). 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 G_r by simply increasing interaction with users. Thus, the volume of sent notifications in (11) is also tracked to balance this effect. A well-behaved agent should have 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.

4 RESULTS & DISCUSSION

Experiments were performed for all combinations of the described initialization methods, algorithms, states, and rewards. For each, 3 tests were executed 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 outliers. All graphs presented show the average result among tested users, employing the N_v and G_r metrics. The standard deviation was also analyzed and is likewise depicted in the displayed graphs. Furthermore, in the tables shown throughout this section, the average G_r and N_v values obtained over 300 days of training are presented.

The overview of our results is shown in Table 1. We compute the average values over all simulations of the combinations of initialization methods, states, and rewards in each algorithm are likewise shown.

Table 1: Average values over all combinations of each initialization method and algorithm.

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

4.1 Previously Trained Model

Table 1 shows that the non-adaptive model tends to be less accurate. Compared with the other two initialization methods, this model displays, on average, a higher standard deviation. Nonetheless, it obtains satisfactory results when resorting to the right combination of algorithm, state, and reward, which in this case is: DQN, using state S1 and S3, and rewards R5 and R6, shown in Table 2.

Table 2: Previously Trained - DQN.

DQN	G_r	N_v
S1 with R5	0.996 ± 0.005	1.926 ± 0.095
S1 with R6	0.983 ± 0.012	1.813 ± 0.123
S3 with R5	0.996 ± 0.004	1.931 ± 0.082
S3 with R6	0.992 ± 0.003	1.864 ± 0.121

DQN leverages less complex state representations throughout the training phase and learn generic user preferences better than any other combination. This shows 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). Although not adaptable to new users' routines, a consistently pleasant user experience can still be offered. 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, similar to supvervised learning algorithms.

4.2 Previously Trained Adaptive Model

Table 1 shows that this method provides the overall best performing average results amongst all three initialization techniques since it can be refined as the final user is actively using the application.

Both UCB and TQ present a good balance between our metrics, as shown in Table 3.

Table 3: Previously Trained Adaptive - UCB and TQ.

Algorithm	State/Reward	G_r	N_v
UCB Day	with R3	0.987 ± 0.014	1.721 ± 0.127
	with R5	0.987 ± 0.009	1.716 ± 0.009
TQ	S1 with R5	0.982 ± 0.013	1.907 ± 0.289
	S1 with R6	0.981 ± 0.009	1.949 ± 0.237
	S2 with R5	0.985 ± 0.004	1.945 ± 0.285
	S2 with R6	0.981 ± 0.011	1.922 ± 0.298

Figure 1 shows that N_v converges early on, but G_r takes longer. Thus, 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.

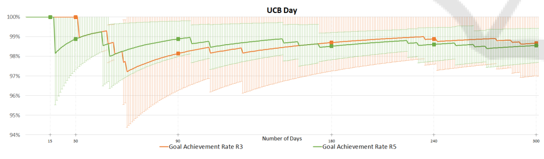
4.3 No Previous Knowledge Model

For this case, simple combinations of both UCB Day and the TQ algorithm, with state representation S1, provide the best results, shown in Table 4. Furthermore, it should be noted that the lower average deviation values are obtained with this implementation, as visible in Table 1.

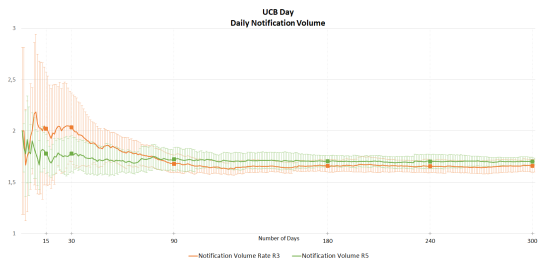
Table 4: No Previous Knowledge - UCB and TQ.

Algorithm	State/Reward	G_r	N_v
UCB Day	with R5	$0.979 \pm 2.8e^{-17}$	$1.714 \pm 2.8e{-}17$
	with R6	0.973 ± 0.004	1.529 ± 0.092
TQ	S1 with R5	0.999 ± 0	2.699 ± 0.633
	S1 with R6	$0.999 \pm 0\ G_r$	2.684 ± 0.660

4.4 Overall Results



(a) Goal Achievement Rate



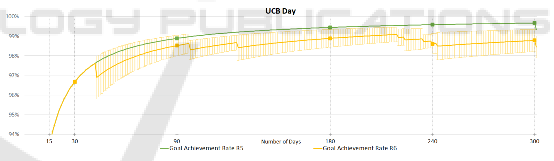
(b) Notification Volume

Figure 1: ComboA: Previously Trained Adaptive, UCB Day - average result over users and across 300 days of training.

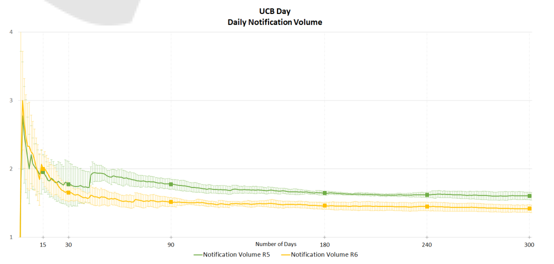
Best Combination - The best performing combination is shown in ComboA - Figure 1. It implements UCB Day, for which a state representation is not required, with reward R5. UCB Day applies an upper confidence bound algorithm for each hour of the day, helping the model to learn what action is better for each decision moment. The initialization method (Previously Trained Adaptive) provides a high G_r rate from the model's deployment, while being able to adapt over time to the user's specific routine, achieving a lower N_v as time goes on. This same combo but with reward R3 provides similar results. In contrast, R6 produced a worse user experience due to the higher penalization value for unanswered notifications, leading to lower N_v and lower G_r.

Similar results, visible in ComboB - Figure 2, 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 the initially discussed combination.

Note that although not providing the best performance, the combination that offered acceptable results consistently throughout all initialization options was implemented with TQ, state S1, and rewards R5 and R6. This affirmation is supported by Table 1. Additionally, DQN presented the most unstable and inconsistent effects, which is also reflected in Table 1, providing an overall less pleasant user experience.



(a) Goal Achievement Rate



(b) Notification Volume

Figure 2: ComboB: No Previous Knowledge, UCB Day - average result among tested users over 300 days of training.

Best State Representation - S1 and S3 tend to perform better throughout all combinations, revealing that more complex, detailed states are not necessarily always more efficient. However, it is not clear if S3 (average day) is better than S1 (average week). **Best Reward Structure -** The reward which generated a better overall performance in the tackled problem was R5. These values provided the best balance between achieving one daily answered notification and not bothering the user.

5 CONCLUSIONS & FUTURE WORK

This work aims 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. We studied a set of RL algorithms, to determine the most desirable combination of initialization method, algorithm, state, and reward definition. 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.

Future work should consider more detailed user reactions that may not be instantaneous, but could arrive within a predefined interval. These responses could be further elaborated by, for example, introducing oblivious dismissal (notification goes unnoticed) and intentional dismissal (people decide not to address it). Lastly, this study relies on a simulator of the user responses, which requires actual testing on mobile devices utilized by real users with different lifestyles, diseases, contexts, and demographics.

ACKNOWLEDGEMENTS

This work has been partially funded by the project LARSyS - FCT Project UIDB/50009/2020 and the project and by the project IntelligentCare – Intelligent Multimorbidity Managment System (Reference LISBOA-01-0247-FEDER-045948), which is co-financed by the ERDF – European Regional Develpment Fund through the Lisbon Portugal Regional Operational Program – LISBOA 2020 and by the Portuguese Foundation for Science and Technology – FCT under CMU Portugal.

REFERENCES

Cornet, V. P. and Holden, R. J. (2018). Systematic review of smartphone-based passive sensing for health and wellbeing. *Journal of Biomedical Informatics*, 77:120–132.

Fiordelli, M., Diviani, N., and Schulz, P. J. (2013). Mapping mhealth research: A decade of evolution. *JMIR*, 15(5):1–14.

Fukazawa, Y., Yamamoto, N., Hamatani, T., Ochiai, K., Uchiyama, A., and Ohta, K. (2020). Smartphone-based mental state estimation: A survey from a machine learning perspective. *JIP*, 28:16–30.

Geneva: WHO (2020). Noncommunicable diseases progress monitor 2020.

Ho, B. J., Balaji, B., Koseoglu, M., and Srivastava, M. (2018). Nurture: Notifying users at the right time using reinforcement learning. *UBICOMP*, pages 1194–1201.

Lin, L.-j. (1993). *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis.

Mehrotra, A. and Musolesi, M. (2017). Intelligent Notification Systems: A Survey of the State of Art and Research Challenges. 1(1):1–26.

Mehrotra, A., Musolesi, M., Hendley, R., and Pejovic, V. (2015). Designing content-driven intelligent notification mechanisms for mobile applications. *UBICOMP*, pages 813–824.

Mehrotra, A., Pejovic, V., Vermeulen, J., and Hendley, R. (2016). My phone and me: Understanding people's receptivity to mobile notifications. *CHI*, pages 1021–1032.

Mnih, V., Kavukcuoglu, K., and Silver (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Rachad, T. and Idri, A. (2020). Intelligent Mobile Applications: A Systematic Mapping Study. *Mobile Information Systems*, 2020.

Richard S. Sutton, A. G. B. (2018). *Reinforcement learning : an introduction*. MIT Press, 2^{o} edition.

Vaizman, Y., Ellis, K., and Lanckriet, G. (2017). Recognizing Detailed Human Context in the Wild from Smartphones and Smartwatches. *IEEE Pervasive Computing*, 16(4):62–74.

Vishwanath, S., Vaidya, K., and Nawal, R. (2012). Touching lives through mobile health-Assessment of the global market opportunity. *PwC*.

Watkins,, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.

WHO Global Observatory for eHealth (2011). mHealth: new horizons for health through mobile technologies: second global survey on eHealth.