Machine Learning Approach to Manage Adaptive Push Notifications for Improving User Experience

Adithya Madhusoodanan
adithya.madhusoodanan3@gmail.com
Department of Electrical and Electronics Engineering
National Institute of Technology Karnataka, India

Anand Kumar M
m_anandkumar@nitk.edu.in
Department of Information Technology
National Institute of Technology Karnataka, India

Kieran Fraser
kieran.fraser@adaptcentre.ie
School of Computer Science and Statistics
ADAPT Centre, Trinity College Dublin, Ireland

Bilal Yousuf
byousuf@scss.tcd.ie
School of Computer Science and Statistics
ADAPT Centre, Trinity College Dublin, Ireland

ABSTRACT

In this modern connected world mobile phone users receive a lot of notifications. Many of the notifications are useful but several cause unwanted distractions and stress. Managing notifications is a challenging task with the large influx of notifications users receive on a daily basis. This paper proposes a machine learning approach for notification management based upon the context of the user and his/her interactions with the mobile device. Since the proposed idea is to generate personalised notifications there is no ground truth data hence performance metrics such as accuracy cannot be used. The proposed solution measures the diversity score, the click through rate score and the enticement score.

CCS CONCEPTS

• **Machine learning → Decision tree classifier**; *Random forest classifier*; **Notification management**.

KEYWORDS

datasets, random forest classifier, decision tree, notifications

ACM Reference Format:
Adithya Madhusoodanan, Anand Kumar M, Kieran Fraser, and Bilal Yousuf. 2020. Machine Learning Approach to Manage Adaptive Push Notifications for Improving User Experience. In *International Workshop on Artificial Intelligence for Mobile and Ubiquitous Communication System.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Mobile phone usage is at a scale never witnessed before. In addition, many mobile applications are widely available for users to install on their devices which creates a large amount of push notifications. These notifications may lead to an overwhelming amount of data left to the user to deal with, with many of them being irrelevant to the user at the time of delivery. Hence a solution to manage notifications on behalf of the user is essential [7]. The research reported in this paper uses the dataset provided by the EvalUMAP Challenge 2020 [1].

This paper reports on the creation of a notification management system which takes in the context of the user and predicts optimal notifications to be shown to the user using machine learning techniques [9]. A synthetic dataset derived from notification engagements captured in-the-wild is used to train the machine learning model.

2020, December–07 – 09, 2020, EAI MobiQuitous 2020
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06...$15.00
https://doi.org/10.1145/1122445.1122456

2 BACKGROUND AND RELATED WORK

A significant challenge faced by researchers in the field of efficient notification management is the lack of open source data. Thus much effort by researchers has been put into the development of systems that could capture data, as existing data in most of the cases are not made available due to ethical reasons [8]. Hence simulated datasets have been applied in a number of recent studies. This work also uses a simulated dataset and applies machine learning to make effective predictions. Machine learning is based on the ability of algorithms to learn and make predictions from a given set of inputs known as a training dataset [11]. There are two main types of machine learning algorithms: supervised learning and unsupervised learning. Here we compare different supervised learning algorithms trained using a simulated dataset.

Corno et al. [2] used a combination of authentic and simulated data to train a variety of machine learning algorithms to verify the potential of machine learning in the field of notification management. In their approach, the context information was predicted by extracting information from connected IoT devices at a given time. The target device for the notification and the time at which the notification was to be shown was also predicted.

Qin et al. on the other hand proposed [4] a machine learning model which identified the relationship between a user's current context and a Do Not Disturb status. Data from a variety of sensors was used to identify user activity (for example: sleeping, walking etc.) and for deciding whether a user was available to view a notification or not. This was very relevant as users may not always need to be instantly interrupted by a notification (e.g. while sleeping) but may need to be alerted at the next period of activity transition (e.g. waking up).

Sarker et al. [11] proposed a method in which noisy instances in the training dataset were eliminated using dynamic noise threshold and implementing a Naive Bayes classifier and Laplace estimator on a per-user basis. This noise-free dataset was then used to train a Decision Tree classifier which was used as the prediction model. Mehrotra et al. [12] proposed a method in which the notifications were grouped according to the applications that initiated them, and the social relationship between the sender and the receiver. Then by using the content and context information, classifiers were trained. It was also reported that such classifiers gave more accurate predictions of user interruptibility.

Pradhan et al. [13] suggested a method in which a machine learning model was used to show only notifications that were important and delayed presentation of the notifications that were unimportant. This is an effective approach as the notifications that may have been irrelevant at a particular time were not discarded indefinitely, thus providing value at later moments which were, perhaps, more contextually relevant. The metrics used for evaluating the model were accuracy, precision, recall and f1-score.

3 EVALUMAP DATASET

The dataset we have used in this work is provided by the EvalUMAP Challenge 2020 [1]. The training dataset consisted of 3 months historical notification engagement data. It is a synthetic dataset containing 9,313 notifications and paired contexts. The synthetic dataset was derived from a real dataset composed of notification engagements collected in-the-wild using an Android application. The original dataset contained data points on over 30,000 notifications, 4,940 smartphone usage logs and 291 questionnaire responses. A Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) was trained to simulate realistic, but synthetic, notification and context parameters for each user. The synthetic data was then divided into three subsets: notifications, contexts and engagements. The dataset is described in Tables 1 and 2. As the name suggests, the context dataset consists of information regarding the context in which the notification was shown. The context data was used to predict the notification to be shown. The context data was primarily used as the feature for our proposed model and the notification data as the output label of the model. The engagement subset was the feed back from the user specifying whether the notification was opened or not.

An Open-AI Gym environment called Gym-push, described in Conlan et al. [1], was used for obtaining the training dataset as well as for evaluating the predicted optimal notifications generated. During training, the Gym-push environment returned 9,313 notifications, contexts and associated engagements. Once training was complete, a test set of contexts was provided by the Gym-push environment for generating an associated set of predicted optimal notifications. These notifications were passed back to Gym-push for evaluation. The notifications were evaluated using three metrics described as follows:

(1) Click-through Rate (CTR) - this metric was used to measure subsequent engagements (open/dismiss) users may take on predicted notifications in the given contexts. Higher CTRs would indicate that the majority of notifications were relevant and valuable in the contexts in which they were presented. A number of classifiers (Adaboost, Naive Bayes, Decision Tree) used to simulate user actions (open/dismiss), were implemented within the Gym-push environment.
(2) Enticement - this is a sanity metric included to ensure an algorithm would not exploit a user for the purposes of maximising the CTR. In a real-world application, a system which generates and pushes notifications could potentially manipulate a subscriber into opening an alert e.g. by sending clickbait. Therefore, ideally, enticement in the predicted set of optimal notifications would be neutral, thus indicating an equilibrium was achieved between enticing a positive notification engagement and respecting the subscriber's right to dismiss.
(3) Diversity - this metric expresses how diverse the set of predicted optimal notifications are. In a real-world scenario, notification subscribers receive notifications spanning a wide range of topics from a wide range sources. This metric ensured the algorithm didn't exploit a particular notification and constantly recommend its use, as this would not reflect reality (a constantly repeated notification would not be tolerated by a user).

4 PREDICTING ADAPTIVE PUSH NOTIFICATIONS

Since the purpose of this work is to predict what notifications are to be shown from a given set of notifications, the task is primarily classification. Hence we initially trained a Random Forest classifier [6] and a Decision Tree classifier [5]. A large number of repetitions in the dataset led us to use a hierarchical learning model. The dataset was divided into different parts and a model was trained by concatenating different parts of the data. The dataset consisted of a number of categorical variables which was encoded using ordinal encoding.

Using the context dataset as the features and the notification dataset as the labels, a Random Forest classifier was trained. A Decision Tree classifier was also trained with the same dataset. These models were used to predict the notifications to be shown to a user based on a given context.

The custom hierarchical model was constructed and trained in a manner given below. A Decision Tree classifier was trained using all the attributes of the contexts subset and was used to predict the visibility, enticement and sentiment features of the training dataset. Here, the *visibility*, *enticement* and *sentiment* features were chosen because they had the highest number of repetitions. The context dataset was concatenated with the *visibility*, *enticement* and *sentiment* features and a Decision Tree classifier was trained with this concatenated dataset to predict the values of *category*, *ledARGB*, *priority*, and *vibrate*. Following this, the features *category*, *ledARGB*, *priority*, and *vibrate* were concatenated with the features of the first Decision Tree to train another Decision Tree with output labels: *app* and *subject*. Finally, the predictions of each of these Decision Tree classifiers were concatenated in order to produce the final output which was a predicted set of notifications. The flow chart representing the model is shown in Figure 1. So for a new context

Table 1: Context

Feature	Explanation
timeOfDay	The time of day split into buckets, possible values: ’morning’, ’afternoon’, ’evening’, ’night’
dayOfWeek	The day of week
unlockCount_prev2	The number of times the user has unlocked their device in the preceding two hours
uniqueAppsLaunched_prev2	The number of unique apps the user has opened in the preceding two hours
dayOfMonth	The day of the month

Table 2: Notification

Feature	Explanation
appPackage	The app that sent the notification
category	The category of the notification, possible values include: ’msg’ and ’email’
ledARGB	The color the LED flashes when a notification is delivered
priority	The priority level of the notification set by the sender
vibrate	The vibration pattern which alerts the user of the notification on arrival
visibility	The visibility level of the notification, possible values include: ’secret’, ’private’ and ’public’
subject	The subject the notification text content was inferred to be
enticement	The enticement value the notification text content was inferred to have
sentiment	The sentiment value the notification text content was inferred to have

Table 3: Comparison between different models

Model	Click through rate score	Diversity Score	Enticement Score
Random Forest Classifier	63.33%	92.63%	32.41%
Decision Tree Classifier	55.92%	98.95%	39.11%
Proposed Model	62.94%	96.84%	41.46%

the Decision Tree 1 will give the *visibility*, *enticement* and *sentiment* . This predicted value along with the context is used as the input for Decision Tree 2 which will give *category*, *ledARGB*, *priority*, and *vibrate* as the output. Now these values are concatenated and given as input for the Decision Tree 3 which gives *app* and *subject* as the output.

The flow chart representing the hierarchical model is given in Figure 1. As illustrated in the chart, **Decision Tree 1** is trained using the context data as input features and the *visibility*, *enticement* and *sentiment* as output labels. Following this, the context data is then concatenated with the *visibility*, *enticement* and *sentiment* features to give **Concatenated Dataset 1**. The **Concatenated Dataset 1** is used as input for training **Decision Tree 2** to predict the *category*, *ledARGB*, *priority*, and *vibrate* features of the notification. Following this, **Concatenated Dataset 1** is concatenated with the *category*, *ledARGB*, *priority*, and *vibrate* features of the notification dataset to create the **Concatenated Dataset 2**, as shown in the diagram. **Concatenated Dataset 2** is then used as an input feature while the *app* and *subject* features of the notification dataset are used as output labels to train **Decision Tree 3**. Hence, for a given context, **Decision Tree 1** will predict the *visibility*, *enticement* and *sentiment* [Prediction 1], **Decision Tree 2** will predict the *category*, *ledARGB*, *priority*, and *vibrate* [Prediction 2] and **Decision Tree 3** will predict the *app* and the *subject* [Prediction 3]. Thus, concatenating Prediction 1, Prediction 2 and Prediction 3 will give a predicted optimal notification for a given context.

As the output of the model is a personalised notification, no ground truth label exists with which to create standard machine learning evaluation metrics - this is an open issue in the adaptive user modeling domain, as comparative evaluations are difficult to

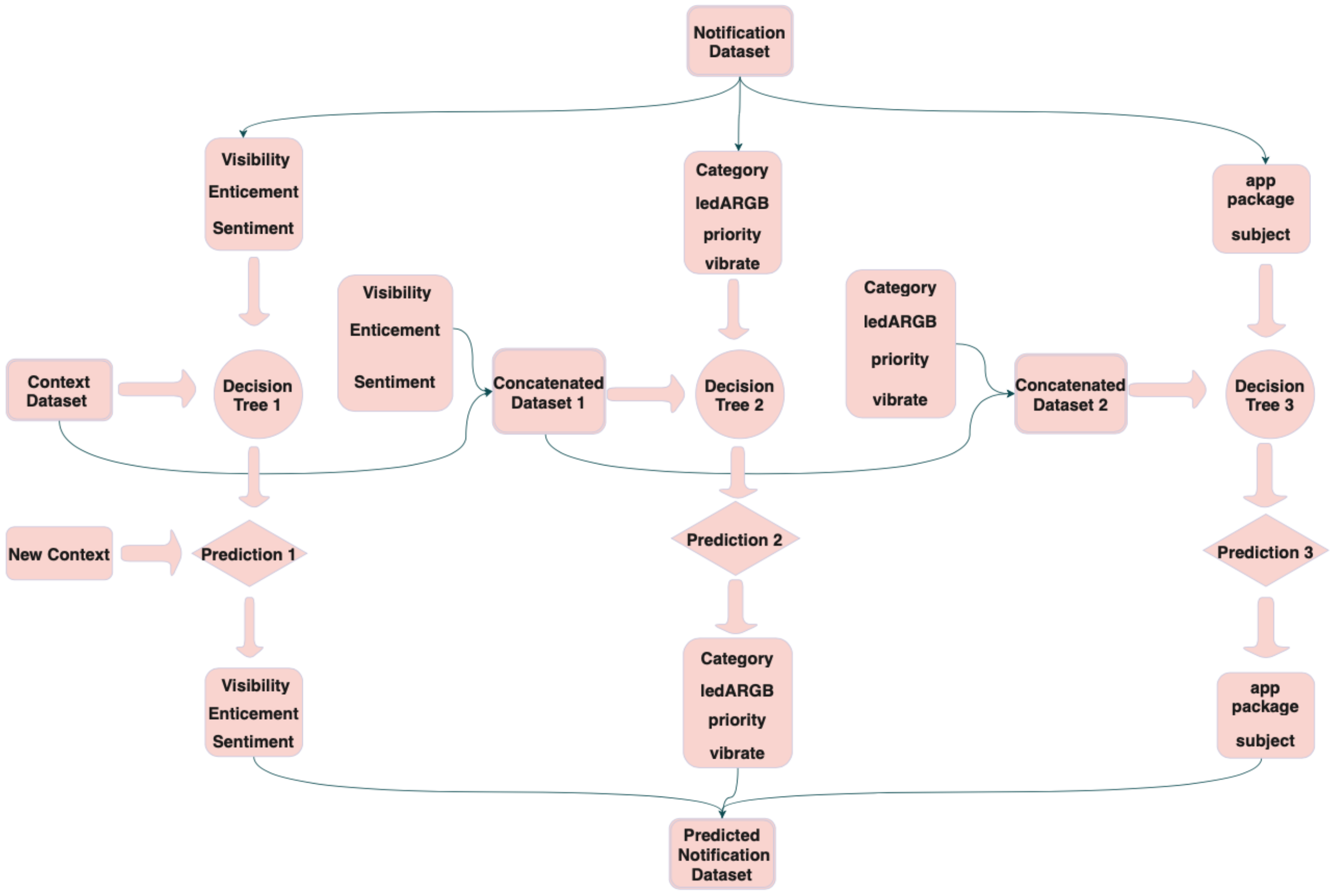
Notification Dataset
Visibility Enticement Sentiment
Category ledARGB priority vibrate
app package subject
Visibility Enticement Sentiment
Category ledARGB priority vibrate
Context Dataset
Decision Tree 1
Concatenated Dataset 1
Decision Tree 2
Concatenated Dataset 2
Decision Tree 3
New Context
Prediction 1
Prediction 2
Prediction 3
Visibility Enticement Sentiment
Category ledARGB priority vibrate
app package subject
Predicted Notification Dataset

Figure 1: Hierarchical Model Architecture

achieve - the EvalUMAP workshop series attempts to address this issue. As an alternative, measures such as diversity, CTR and enticement score have been defined in place of standard ML metrics for comparative evaluation. The scores achieved by the model proposed in this paper are presented in Table 3.

5 EXPERIMENTS AND RESULTS

For evaluating the performance of the model the OpenAI Gym environment described in [1] was used. Gym-push used three classifiers (Adaboost, Naive Bayes, Decision Tree), trained on historical notification engagement data (i.e. not used for training, validation or testing) of users, to simulate engagement (open/dismiss) actions for notifications in a given context. The optimal notifications predicted by the proposed model were passed to the Gym-push environment, in which scores for CTR, enticement and diversity were calculated and returned for assessment and tuning.

A comparison between the three models evaluated, using held-back test data provided by Gym-push, are illustrated in Table 3. The models were compared on the basis of CTR, diversity and enticement. The custom hierarchical model achieved an enticement score closer to equilibrium (50%) than the other algorithms evaluated meaning it learned to optimise CTR using enticement but without exploiting the end-user. The proposed model also achieved high diversity and CTR scores, beaten only by the Random Forest. An argument can be made for choosing the proposed model as the higher enticement score demonstrates that more interesting notifications were delivered, as opposed to the quite low enticing notifications delivered by the Random Forest classifier - thus the proposed model may offer an improved user-experience. Fluctuations in enticement over validation epochs can be seen in Figure 2a. The individual CTR scores simulated by the three differing classifiers implemented within the Gym-push environment during training are illustrated in Figure 2b. The overall CTR, enticement and diversity scores produced during the training of the proposed model are shown in Figures 2e, 2c and 2d respectively.

6 CONCLUSION AND FUTURE SCOPE

This paper proposed the effective management of notifications using a hierarchical model. The proposed model was found to outperform conventional classification algorithms such as Random Forest and Decision Tree by producing a better diversity, Click-Through Rate and enticement score. Future work will be devoted to training a neural network with the synthetic data shared by Gym-push in order to carry out a comparative evaluation between

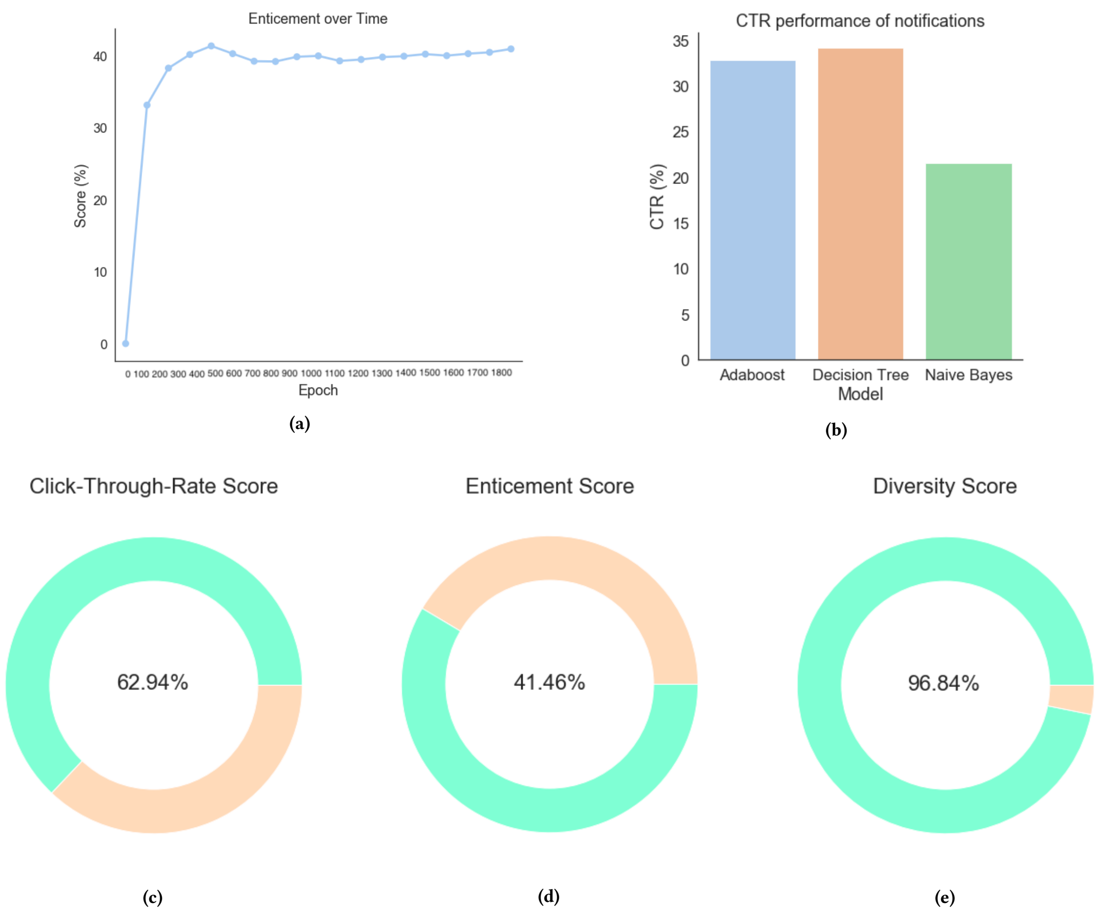

Figure 2: Results of proposed model as evaluated by Gym-push.

Deep Learning methods and the model proposed in this paper. Additionally, other researchers in the space are encouraged to evaluate their own models using the Gym-push environment. In addition, the engagement data shared within Gym-push could also be used for future development of Reinforcement Learning algorithms, as demonstrated in [14], which could further personalise and improve on the predictions of optimal notifications created for individual users based on their contexts and past engagement behaviours.

ACKNOWLEDGEMENT

This research is supported by the National Institute of Technology Karnataka, Surathkal, India under student internship fund. It is also supported by the Science Foundation Ireland through the CNGL program (Grant 12/CE/I2267) in the ADAPT Center (www.adaptcentre.ie) at Trinity College Dublin.

REFERENCES

[1] Conlan, Owen and Fraser, Kieran and Kelly, Liadh and Yousuf, Bilal (2019) *A User Modeling Shared Challenge Proposal. In Proceedings of the 10th Conference and Labs of the Evaluation Forum, Lecture Notes in Computer Science (LNCS)* Springer, September 2019.

[2] Fulvio Corno, Luigi De Russis, and Teodoro Montanaro. 2015. *A context and user aware smart notification system. In Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on.* IEEE, 645–651.

[3] P. Harrington, Machine Learning in Action. Manning Publications Company, 2011.

[4] Y. Qin, T. Bhattacharya, L. Kulik and J. Bailey, *"A Context-aware Do- not-disturb Service for Mobile Devices," in Proceedings of the 13th International Conference on Mobile and Ubiquitous Multimedia,* New York, NY, USA, 2014.

[5] Patel, Harsh & Prajapati, Purvi. (2018). Study and Analysis of Decision Tree Based Classification Algorithms. International Journal of Computer Sciences and Engineering. 6. 74-78. 10.26438/ijcse/v6i10.7478.

[6] Ali, Jehad & Khan, Rehanullah & Ahmad, Nasir & Maqsood, Imran. (2012). Random Forests and Decision Trees. International Journal of Computer Science Issues(IJCSI). 9.

[7] Fraser, Kieran & Yousuf, Bilal & Conlan, Owen. (2016). A context-aware, info-bead and fuzzy inference approach to notification management. 1-7. 10.1109/UEMCON.2016.7777832.

[8] Fraser, Kieran & Yousuf, Bilal & Conlan, Owen. (2017). Synthesis & Evaluation of a Mobile Notification Dataset. 179-184. 10.1145/3099023.3099046.

[9] Mehrotra, Abhinav & Hendley, Robert & Musolesi, Mirco. (2017). Interpretable Machine Learning for Mobile Notification Management: An Overview of PrefMiner. GetMobile: Mobile Computing and Communications. 21. 35-38. 10.1145/3131214.3131225.

[10] Corno, F., De Russis, L., & Montanaro, T. (2015). A context and user aware smart notification system. 2015 IEEE 2nd World Forum on Internet of Things (WF-IoT). doi:10.1109/wf-iot.2015.7389130

[11] Sarker, I. H. (2019). A Machine Learning based Robust Prediction Model for Real-life Mobile Phone Data. Internet of Things. doi:10.1016/j.iot.2019.01.007

[12] Abhinav Mehrotra, Mirco Musolesi, Robert Hendley, and Veljko Pejovic. 2015. Designing content-driven intelligent notification mechanisms for mobile applications. In Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '15). Association for Computing Machinery,

New York, NY, USA, 813–824. DOI:https://doi.org/10.1145/2750858.2807544

[13] S. Pradhan, L. Qiu, A. Parate and K. Kim, "Understanding and managing notifications," IEEE INFOCOM 2017 - IEEE Conference on Computer Communications, Atlanta, GA, 2017, pp. 1-9, doi: 10.1109/INFOCOM.2017.8057231.

[14] Rowan Sutton, Kieran Fraser, and Owen Conlan. 2019. A Reinforcement Learning and Synthetic Data Approach to Mobile Notification Management. In Proceedings of the 17th International Conference on Advances in Mobile Computing & Multimedia (MoMM2019). Association for Computing Machinery, New York, NY, USA, 155–164. DOI:https://doi.org/10.1145/3365921.3365932

[15] Bo-Jhang Ho, Bharathan Balaji, Mehmet Koseoglu, and Mani Srivastava. 2018. Nurture: Notifying Users at the Right Time Using Reinforcement Learning. In Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers (UbiComp '18). Association for Computing Machinery, New York, NY, USA, 1194–1201. DOI:https://doi.org/10.1145/3267305.3274107

[16] Shamsi T. Iqbal and Brian P. Bailey. 2008. Effects of intelligent notification management on users and their tasks. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08). Association for Computing Machinery, New York, NY, USA, 93–102. DOI:https://doi.org/10.1145/1357054.1357070