Generation and Evaluation of Personalised Push-Notifications

Kieran Fraser
kieran.fraser@adaptcentre.ie
ADAPT Centre, Trinity College
Dublin, Ireland

Bilal Yousuf
byousuf@scss.tcd.ie
ADAPT Centre, Trinity College
Dublin, Ireland

Owen Conlan
owen.conlan@scss.tcd.ie
ADAPT Centre, Trinity College
Dublin, Ireland

ABSTRACT

A shared challenge in the domain of User Modeling, Adaptation and Personalisation is proposed for the 2019 EvalUMAP workshop whereby the evaluation of user models generating personalised push-notifications is to be explored. As such, this paper presents a description of the evaluation process, a solution to the first proposed challenge and details the results obtained from the *Gym-Push* evaluation environment.

CCS CONCEPTS

• **Human-centered computing** → **User models**; *Usability testing*; • **Information systems** → *Open source software*.

KEYWORDS

evaluation, personalisation, user modeling, push-notifications

ACM Reference Format:
Kieran Fraser, Bilal Yousuf, and Owen Conlan. 2018. Generation and Evaluation of Personalised Push-Notifications. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The task proposed in the 2019 EvalUMAP white paper [6] is the generation of personalised push-notifications. This use-case is meant to support the creation and subsequent evaluation of user modeling, adaptation and personalisation techniques in a standard way such that comparisons and benchmarks can be drawn. The task is comprised of two challenges, both of which provide a data set from which a personalised model can be created. The data set consists of a number of synthetic and contextual smartphone and push-notification features derived from a 3 month long in-situ smartphone usage study. This paper will focus on Challenge 1.

1.1 Challenge 1

1.1.1 Description. Three months of contextual push-notification data was provided from a number of distinct users. The goal of the challenge was to create a user model capable of generating personalised push-notifications, given a particular context such as the place or activity of the user. The applicability of this challenge to the real world is quite tangible as actionable AI is moving ever closer to edge technologies, such as the mobile device. Subsequently, a user model created for a challenge such as this could be adapted to create personalised push-notifications, in real-time, on a user's device instead of being created and pushed by application owner's in the cloud.

1.1.2 Evaluation. The performance of the model is evaluated using a custom *OpenAI Gym* [1] environment, *Gym-Push* [3], which contains an additional three months data on which the user model is evaluated. Evaluation is achieved by installing *Gym-Push* and querying the *push_eval_1-vX* environment for a user's context data. The context data can then be fed into a user model to generate personalised push-notifications dependent on the context. These newly generated notifications are then passed back to the gym environment via the *testNotifications()* method which begins the evaluation of the model output.

1.1.3 Metrics. The metrics used in the evaluation of Challenge 1 are *performance* and *response time*. The *performance* metric describes the degree of notification engagement improvement over original notifications pushed at a user. The *response time* metric describes how long it takes for a model to generate a set of notifications given the context as seed, as a real world use of this task would be to generate notifications in real-time. The implementation of these metrics are further outlined in following sections.

2 METHOD

The method proposed by this paper for Challenge 1 is to use a *Conditional* Generative Adversarial Network (GAN) [2] to learn a user's habits with regard their notification engagements with respect to differing contexts. GAN's are a set of neural networks which can be used for generating synthetic data which mimics the real world data used to train it. The GAN is made up of a generator and discriminator network. The generator takes random noise as input and outputs a notification. The discriminator network alternates between a real notification sample and a synthetic notification sample as input and as output indicates if the notification is synthetic or not. When trained together, both networks attempt to better the other - the discriminator always trying to correctly categorize a notification as synthetic or real and the generator always trying to pass off synthetic samples as real samples. Ideally, the cross-entropy loss of both the discriminator and generator should converge such that the generator produces sufficiently real notification samples which cannot be deciphered as synthetic by the discriminator.

3 NOTIFICATION PERSONALISATION

Personalised push-notifications were generated for each user using a *Conditional Wasserstein* GAN. The context data features were used as the condition on which to train the GAN. This enabled the creation of a generative model which could take as input, a number

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

Table 1: Comparison of original (real) notifications and personalised (synth) notifications generated for first 5 users.

	Total Notifications		Unique Apps		Unique Subjects	
User Id	Real	Synth	Real	Synth	Real	Synth
1	6963	7404	23	8	12	1
2	1632	1666	8	3	6	2
3	3245	3237	3	1	4	1
4	1959	1771	8	4	2	1
5	5031	5591	12	4	17	2

of contextual features, and produce a push-notification as output. The action of the notification taken by the user was also added as a conditional feature which allowed for the additional option of generating only notifications which were habitually opened by the user in the given context, thus ensuring the notifications generated were items which the user was receptive to.

3.0.1 Implementation. A Multilayer Perceptron (MLP) was used for both generator and discriminator. The dimension of input used for the generator was 10 and latent space values were sampled from a uniform distribution. The notifications were encoded (one-hot) into multidimensional vectors of length 28 and used as input for the discriminator. The networks were trained using *Root Mean Square Propogation* (RMSProp) in 128*mini-batch* chunks over 2000 epochs.

For each user in the training data provided in Challenge 1, a generative model was created using 3 months of their notification engagement data. Once trained, the *Gym-Push* environment was queried for context. The context was used to seed the generation of a further 3 months of notification engagement data (Table 1), which was passed back to the environment for evaluation.

4 EVALUATION

The *performance* of the personalised notifications was calculated by training a classifier to take the place of the user and act (open/dismiss) upon the newly generated personalised notifications. Each classifier was trained on 3 months of historical user notification engagements. Scikit-learn's [5] implementation of the AdaBoost classifier was used in this case. The *Click Through Rate* (CTR) was used as the performance metric of choice as it is an industry standard for measuring notification engagement performance. The CTR is the number of opened notifications over the total number of notifications pushed at the user. Figure 1 highlights the performance of the personalised notifications compared with the original actions taken by the user and also compared with the actions predicted by the classifier for the original notifications (which is mimicking the user). Discrepancies in performance between the original and classifier actions are expected, as when trained, the user classifiers do not reach 100% accuracy in mimicking the user. More importantly however, the personalised notifications result in higher CTR's in 6 of the 10 users and in some cases (e.g. user 1), surpass the CTR of original actions taken by the user.

The *response time* of the model is defined as the time taken for the model to generate the personalised notifications of a user, given

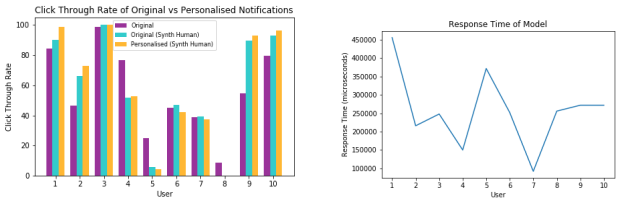

Figure 1:
Left: Comparison of results: 1) Original: CTR of original notifications taken by real user; 2) Synth Human: CTR of original notifications acted upon by classifier; 3) Personalised: CTR of personalised notifications acted upon by classifier. Right: Response time of the model. The time taken for personalised notifications to be generated.

the contextual data. The *Gym-Push* environment calculates the time from when a user's contextual data is queried to when the test method is called. Figure 1 illustrates the response time of the model for each user. Naturally user models with higher quantities of notification training data take longer to respond with personalised notifications as the GAN trains over a larger corpus. This provides a good benchmark on which to compare the efficiency of other models with respect to the real-time notification generation problem.

5 LIMITATIONS & FUTURE WORK

The evaluation of the personalised models is limited as an imperfect classifier is replacing a human in the evaluation of the personalised notifications. It does however, provide an intermediate interface through which performance benchmarks can be set and models tested against before carrying out in-situ studies. Additional metrics will be explored for use in future evaluations, such as: diversity, coverage, novelty and serendipity [4]. Additional synthetic data sets will also be added to the *Gym-Push* environment for both the *Personalisation* and *Reinforcement Learning* communities.

ACKNOWLEDGMENTS

This research is supported by the ADAPT Centre for Digital Content Technology under the SFI Research Centre Program (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

REFERENCES

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:arXiv:1606.01540
[2] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. 2017. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*. 5767–5777.
[3] Gym-Push 2019. A custom OpenAI Gym environment. Retrieved March 20, 2019 from https://github.com/kieranfraser/gym-push
[4] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems. *ACM Trans. Interact. Intell. Syst.* 7, 1, Article 2 (Dec. 2016), 42 pages. https://doi.org/10.1145/2926720
[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[6] Proposal for a Shared Challenge in the UMAP Space 2019. Adapt, EvalUMAP 2019. Retrieved March 20, 2019 from http://evalumap.adaptcentre.ie/