Deep Sequential Recommendation for Personalized Adaptive User Interfaces

Harold Soh, Scott Sanner, Madeleine White, Greg Jamieson
University of Toronto
harold.soh@utoronto.ca, ssanner@mie.utoronto.ca, maddy.white@mail.utoronto.ca,
jamieson@mie.utoronto.ca

ABSTRACT

Adaptive user-interfaces (AUIs) can enhance the usability of complex software by providing real-time contextual adaptation and assistance. Ideally, AUIs should be personalized and versatile, i.e., able to adapt to each user who may perform a variety of complex tasks. But this is difficult to achieve with many interaction elements when data-per-user is sparse. In this paper, we propose an architecture for personalized AUIs that leverages upon developments in (1) deep learning, particularly gated recurrent units, to efficiently learn user interaction patterns, (2) collaborative filtering techniques that enable sharing of data among users, and (3) fast approximate nearest-neighbor methods in Euclidean spaces for quick UI control and/or content recommendations. Specifically, interaction histories are embedded in a learned space along with users and interaction elements; this allows the AUI to query and recommend likely next actions based on similar usage patterns across the user base. In a comparative evaluation on user-interface, web-browsing and e-learning datasets, the deep recurrent neural-network (DRNN) outperforms state-of-the-art tensor-factorization and metric embedding methods.

Author Keywords

Adaptive User Interface; Deep Learning; Personalization

ACM Classification Keywords

I.2.6. Artificial Intelligence: Learning; H.5.2 Information Interfaces and Presentation(e.g. HCI): User Interfaces

INTRODUCTION

Advances in software capabilities and widening access to digital content have dramatically altered what is achievable with today's computing systems. However, growing software complexity can hamper usability and lead to end-user frustration. Adaptive user-interfaces (AUIs) (e.g., [13, 20, 28]) can potentially mitigate these issues by providing real-time contextual adaptation and assistance in the form of recommended actions that are personalized for the current user.

IUI 2017, March 13–16, 2017, Limassol, Cyprus.

http://dx.doi.org/10.1145/3025171.3025207

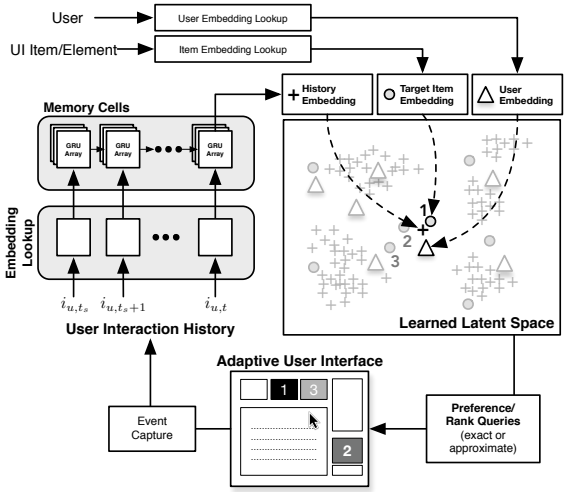

Figure 1. A closed-loop schematic of our proposed personalized adaptive user interface (AUI) which incorporates a deep neural model to learn a latent embedding space. The model uses gated recurrent units (GRUs) to project interaction histories into this space, along with users and target items (UI control and/or content elements). Adaptation is performed by querying this space via nearest-neighbor (NN) methods to obtain recommended items—exact search can be used with small item sets and approximate NN searches permit favorable scaling with item set size. In the example image above, the top 3 elements (labelled 1 through 3) can be adaptively highlighted to ease user interaction, while low rank items can be rendered inconspicuous. As the user interacts with the system, the user interaction history is updated, which subsequently changes the recommendations and resulting adaptations.

In this paper, we contribute an architecture for personalized AUIs that incorporates a deep sequential recommendation model (Fig. 1). Specifically, our approach leverages upon a deep recurrent neural network (DRNN) to learn a latent embedding space that permits data-sharing in a collaborative-filtering fashion. By querying this shared latent space, the AUI is able to exploit similar usage patterns from across the user base to make personalized adaptations/recommendations for relatively new users and novel scenarios.

A key feature is that the DRNN not only embeds users and target items (UI control and content elements), but also long-term interaction histories. This is in contrast to general sequential recommendation methods such as factored personalized Markov chain (FPMC) [22] and metric embedding [8, 27], which only model pairwise, but not higher-order, historical dependencies. The Markovian assumption is generally

inappropriate for AUIs since one user action can be followed by any number of competing actions depending on task and context. Here, our model learns long-term dependencies via gated recurrent memory units (GRUs) [5], which do not aim to model all possible higher-order historical interactions, but instead learns what information to propagate in a linear sequential fashion. A crucial distinction from very recent work applying GRUs to general sequential recommendation [11] is that we also consider Euclidean—rather than typical dot-product—embedding spaces. This allows us to apply fast approximate k-nearest-neighbor (k-NN) search methods that scale favorably with large interaction element sets common in modern UIs.

Experiments on three different datasets involving interactions with user-interface, website, and e-learning system, show that the DRNN outperforms recent Markov models including a tensor factorizer [22] and Euclidean collaborative filtering [18]. Furthermore, we show approximate nearest neighbor searches in our learnt embedding space allows for an order of magnitude decrease in query times—a result particularly relevant for real-time interactivity and adaptation. In the remainder of this paper, we detail our proposed architecture and our experimental findings.

DEEP SEQUENTIAL RECOMMENDATION

In the sequential recommendation task, we aim to suggest promising next items $y = i_{u,t+1} \in I$ given tuples $(u, V_{u,t})$ of users $u \in U$ and interaction (or session) histories $V_{u,t} = (i_{u,t_s}, i_{u,t_s+1}, \dots, i_{u,t})$ consisting of previous items $i_{u,t} \in I$. To achieve this objective, our DRNN model (Fig. 1) learns to embed users, histories and target items into a shared latent space where elements that co-occur more frequently are closer, or more similar. This latent space can then be queried using exact or approximate nearest-neighbor methods to obtain recommended UI elements or content items.

Embedding Symbols as Vectors

At the first component layer, user and item symbols are mapped to real vectors (embeddings). This mapping is essentially a table lookup where each user u is associated with a corresponding vector $\mathbf{x}_u \in \mathbb{R}^{n_u}$. Likewise, each item i has a associated $\mathbf{x}_i \in \mathbb{R}^{n_i}$, allowing the user's interaction history $V_{u,t}$ to be represented as $X_{u,t} = (\mathbf{x}_{u,t_s}, \mathbf{x}_{u,t_s+1}, \dots, \mathbf{x}_{u,t})$. Target items j are embedded as $\mathbf{x}_j \in \mathbb{R}^{n_j}$. The sizes of these vectors, n_u, n_i, and n_j, are model parameters; we set the embedding sizes to be equal $n_u = n_i = n_j = n_{\text{EMB}}$ since the learnt space is shared. While it is straightforward to work with the user and target item embeddings, the interaction history is a tuple of indeterminate length, which complicates the recommendation process. To address this issue, the DRNN compresses the history into a fixed-sized memory state.

Memory: Gated Recurrent Units

The DRNN uses Gated Recurrent Units (GRUs) [5] (Fig. 2), a streamlined variant of long short-term memory [12], that has become popular due to its simplicity and performance [17]. In contrast to the traditional recurrent cell that lacks control over its hidden state, the GRU learns to operate two internal

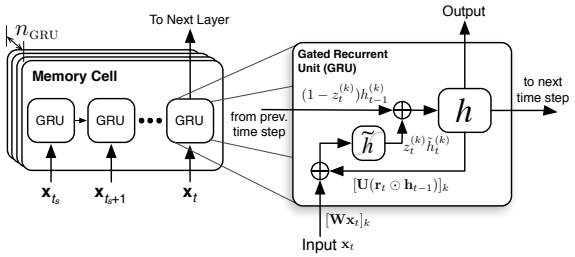

Figure 2. The model's memory is composed of Gated Recurrent Units (GRUs) that selectively update or reset their hidden states via two gates for each cell k: the update gate z_k controls the level at which the internal state h is replaced by a new hidden state $\tilde{h}$ (a weighted average). The reset gate r_k controls how much of the previous hidden state to forget. After a user's interaction history is fed into the cells, the hidden state is relayed to higher layers.

structures—the update and reset gates—that allow it to control what to remember and forget. More formally, a cell k that has state $h_{t-1}^{(k)}$ and receives a new input $\mathbf{x}_t$ is updated via

$$h_t^{(k)} = (1 - z_t^{(k)})h_{t-1}^{(k)} + z_t^{(k)}\tilde{h}_t^{(k)}, \tag{1}$$

i.e., a weighted average of its previous state and a candidate activation $\tilde{h}_t^{(k)}$. This interpolation is controlled by the update gate $z_t^{(k)}$, which relies on two learnt parameters $\mathbf{W}_z$ and $\mathbf{U}_z$,

$$z_t^{(k)} = \text{sigm}\big([\mathbf{W}_z\mathbf{x}_t + \mathbf{U}_z\mathbf{h}_{t-1})]_k\big). \tag{2}$$

The candidate activation $\tilde{h}_t^{(k)}$ is computed via

$$\tilde{h}_t^{(k)} = \tanh([\mathbf{W}\mathbf{x}_t + \mathbf{U}(\mathbf{r}_t \odot \mathbf{h}_{t-1})]_k) \tag{3}$$

where $\odot$ denotes element-wise multiplication. The cell's reset gate $r_t^{(j)} = [\mathbf{r}_t]_k$ is determined by two learnt matrices $\mathbf{W}_r$ and $\mathbf{U}_r$,

$$r_t^{(k)} = \text{sigm}\big([\mathbf{W}_r\mathbf{x}_t + \mathbf{U}_r\mathbf{h}_{t-1})]_k\big) \tag{4}$$

The previous hidden state is effectively dropped when the reset gate's value nears zero, and hence, cells that learn short-term dependencies have active reset gates. In comparison, cells that learn to model long-term dependencies have active update gates [5]. Our model uses an array of n_{GRU} cells that learn to embed the user interaction history $X_{u,t} = (\mathbf{x}_{u,t_s}, \mathbf{x}_{u,t_s+1}, \dots, \mathbf{x}_{u,t})$ as "memory state" $\mathbf{h}_{u,t}$ that is relevant for next item recommendation.

Deriving Preference Scores from Embeddings

For a given user u with memory state $\mathbf{h}_{u,t}$, we can obtain preference scores over target items j, denoted $a_{u,t+1,j}$, by imposing a structure on the latent embedding space. In this work, we consider two models where the preference scores are represented as:

- Dot-products between embedding vectors, $a_{u,t+1,j} = \mathbf{x}_u^\top\mathbf{x}_j + \mathbf{h}_{u,t}^\top\mathbf{x}_j$, which is the approach taken in Markov tensor factorization [22];
- (squared) Euclidean distances, $-a_{u,t+1,i} = \|\mathbf{x}_u - \mathbf{x}_j\|_2^2 + \|\mathbf{h}_{u,t} - \mathbf{x}_j\|_2^2$, similar to recent Euclidean and metric embedding models [18, 4].

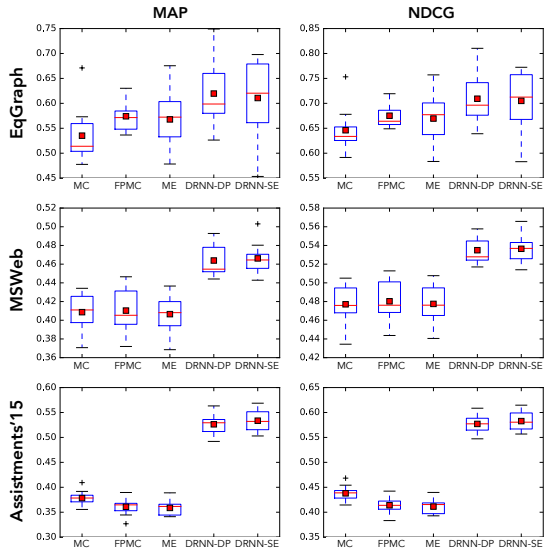

Figure 3. Box-plots of MAP and NDCG scores on the EqGraph, MSWeb and Assistments'15 datasets (mean scores are shown as red squares). On all three datasets, the DRNN achieves the highest scores (significant at the 1% level; Kolmogorov-Smirnov and Mann-Whitney U tests with multiple-testing correction).

Fast Recommendation Queries

Since AUIs are often expected to be real-time interactive, an important consideration is the efficiency of recommendation queries. In typical settings, the AUI provides assistance/adaptations based on the top k items (with largest $a_{u,t+1,j}$). In this respect, the Euclidean space is preferable to dot-product spaces; the search for preferred items reduces to a k-nearest-neighbors (k-NN) search since

$$\begin{aligned} a_{u,t+1,j} &= \|\mathbf{x}_u - \mathbf{x}_j\|_2^2 + \|\mathbf{h}_{u,t} - \mathbf{x}_j\|_2^2 \\ &= \|\hat{\mathbf{x}}_{u,h} - \hat{\mathbf{x}}_j\|_2^2 \end{aligned} \quad (5)$$

where $\hat{\mathbf{x}}_{u,h} = [\mathbf{x}_u; \mathbf{h}_{u,t}]$ (a concatenation of $\mathbf{x}_u$ and $\mathbf{h}_{u,t}$), and $\hat{\mathbf{x}}_j = [\mathbf{x}_j; \mathbf{x}_j]$. Efficient k-NN search methods for Euclidean spaces exist; for low dimensions, the well-known kd-tree is sufficient, and for high dimensions (> 30), approximate methods (e.g., [24]) have been shown to achieve high precision at low computational cost. In this work, we use FLANN [21] to find the top-k recommendations.

It is worth noting that fast NN search in dot-product spaces—known as the maximum inner-product search (MIPS) problem—is an active research area. Key methods, e.g, order-preserving transformations [2], involve a transformation into a Euclidean space, followed by a NN search. Directly embedding the elements into a Euclidean space allows us to skip this transformation step.

Model Training via Candidate Sampling

The entire model—embeddings and GRU cells—can be trained via standard backpropagation under an appropriate loss function. Unfortunately, typical loss functions become computationally infeasible given the large interaction element sets I common in modern user interfaces. To address this issue, we train the DRNN using candidate sampling—the loss is evaluated against a small set of candidate or contrastive classes instead of over I. In this work, we used the recently-proposed sampled softmax used in neural translation [15][1]. With a simple discrete uniform distribution $Q(i|a_{u,t+1,y}) = 1/|I|$ to form the candidate set S, we apply an augmented loss function, $L = \sum_{u,t} l_{\text{SM}}(u,t)$,

$$l_{\text{SM}} = -a_{u,t+1,y} + \log \frac{\sum_{j \in S} \exp(a_{u,t+1,j} - \log |I|)}{|I|} \quad (6)$$

where l_{SM} is loss per user interaction, $a_{u,t+1,y}$ is the preference score given in (5), , and $y = i_{u,t+1}$ is the target item observed at time $t+1$. The loss (6) is composed of two parts where the second part of the RHS corrects for the sampling approximation[2]. With this loss in hand, the DRNN can be optimized using (stochastic) gradient-based methods such as Adam [19].

EXPERIMENTS

In this section, we compare our proposed neural model against existing models for sequential recommendation, specifically a standard Markov Chain (MC), the state-of-the-art Factorized Personalized Markov Chain (FPMC) [22], and Euclidean Metric Embedding (ME) [18]. We implemented two variants of the DRNN with dot product (DRNN-DP) and squared Euclidean (DRNN-SE) embedding spaces.

Datasets: The methods were compared using the Equation Grapher UI dataset (EqGraph) [7], Microsoft Web Data [3] and the 2015 ASSISTments dataset. EqGraph consists of 51 sequences (grouped into 3 different tasks) of user interactions with an equation grapher UI. Each sequence comprises symbols representing one of 16 area-of-interest regions corresponding to UI elements (for more details, see [7]). The latter two datasets contain content items; MSWeb contains per-user resource accesses at microsoft.com for one week, and Assistments'15 contains student interactions with problem sets using the ASSISTments e-learning system [10]. Following [22], users that did not interact with at least 10 items were dropped—the filtered EqGraph contained 51 users, 16 items, and 1277 interactions; MSWeb contained 1205 users, 246 items and 6623 interactions; and Assistments'15 contained 4130 users, 100 items and 51,318 interactions.

Model Training and Parameters: We trained each model for a maximum of 200 epochs; optimization was performed using the Adam algorithm [19] with a learning rate of 10^{-2}, 100-sample mini-batches, and 15 negative samples. For all models, we performed a grid search over embedding sizes $n_{\text{EMB}} \in \{32, 64, 128\}$ and dropout regularization $p_{\text{DROP}} \in \{0.05, 0.10, 0.15, 0.20\}$. For the recurrent neural models, we used a single-layer memory of n_{EMB} GRU cells.

Performance Evaluation: Each algorithm was tasked to predict the next relevant UI element or content item. We performed 10-fold cross-validation where 90% of the sequences

[1]The sampled softmax achieved better scores compared to Noise Contrastive Estimation and the sampled logistic in our preliminary trials. Alternative training losses, e.g., Bayesian Personalized Ranking [22], can be substituted for the sampled softmax depending on application requirements.

[2]Please see [15, 1] for more details, including derivation.

Dataset	MC	FPMC	ME	DRNN-DP	DRNN-SE
EqGraph	0.535	0.574	0.568	0.620	0.611
	(0.054)	(0.031)	(0.057)	(0.062)	(0.078)
MSWeb	0.409	0.410	0.407	0.475	0.466
	(0.020)	(0.023)	(0.021)	(0.022)	(0.016)
Assist'15	0.378	0.361	0.359	0.526	0.534
	(0.015)	(0.016)	(0.014)	(0.020)	(0.022)

Table 1. Mean MAP scores (with standard deviation in brackets) obtained by the compared methods on three datasets. DRNN achieves ∼15-50% higher MAP scores compared to MC, FPMC and ME. Scores between the two DRNN variants were similar.

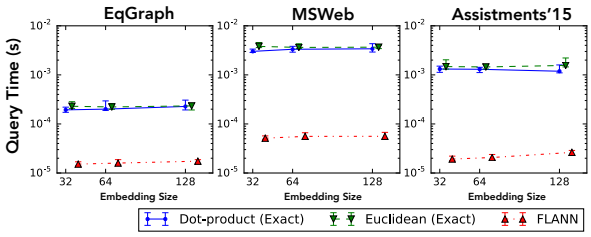

Figure 4. Median preference/rank query times in seconds (with lower and upper quantiles) with increasing embedding size (32, 64, and 128). Approximate NN searches with FLANN were often more than a magnitude faster compared to exact methods.

were used for training and 10% for testing. As performance measures, we used the standard mean average precision (MAP) and normalized discounted cumulative gain (NDCG) at $k = 10$, computed over the final 50% of each test sequence.

Results and Discussion

The performance scores achieved by the compared methods are summarized in Fig. 3 and Tbl. 1. In short, the DRNN variants outperformed the other methods on all three datasets; the observed differences are statistically significant at the $\alpha = 1\%$ level (Kolmogorov-Smirnov and Mann-Whitney U tests with Holm-Sidak corrected p-values). Compared to the Markov models, the recurrent model achieves ∼15-50% higher MAP scores, providing evidence that capturing long-term interactions improves recommendation on sequential tasks. The two DRNN variants performed similarly, indicating that both the dot-product and squared-Euclidean latent spaces were effective for this task.

The recommendation time per query (in seconds) when using FLANN compared to an exact method (a linear distance/dot-product computation followed by a sort) is shown in Fig. 4. Approximate searches in FLANN in the Euclidean space took more than an order of magnitude less time compared to the exact methods, whilst retaining similar accuracy. As expected, computational costs were typically higher with the larger embedding sizes, but at $\sim 10^{-4}$ seconds, queries were well within requirements for real-time interaction with users.

Fig 5 shows DRNN's performance generally improved with embedding size, with a few exceptions where $n_{\text{EMB}} = 128$; in these cases, further optimization or regularization may be required to improve performance scores. On the EqGraph dataset, the DRNN with $n_{\text{EMB}} = 32$ has similar performance

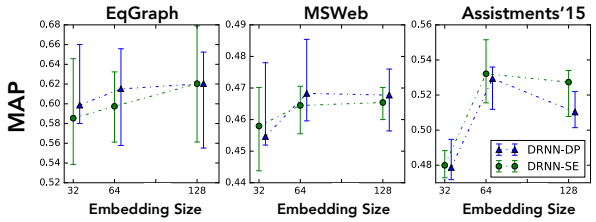

Figure 5. Median MAP scores (with lower and upper quantiles) versus increasing embedding size (32, 64, and 128) for the DRNN models. The best performing embedding sizes (in terms of MAP) were 128 for EqGraph and 64 for MSWeb and Assistments'15.

to FPMC and ME, showing that a sufficiently large embedding/memory size is needed to obtain the positive effects of longer histories.

In practice, desired accuracy should be balanced against the increased response times associated with prediction using more complex models. Previous HCI studies have shown that both these factors have significant effects; system response times strongly influence user satisfaction [23] and engagement [26], and accuracy impacts user trust and reliance, essential for encouraging continued use [6]. Relative to the compared methods, the DRNN-SE provided higher (or similar) accuracies, with fast prediction.

SUMMARY AND CONCLUSIONS

This paper presented an architecture for an adaptive user-interface with a RNN that performs sequential recommendation of content and control elements. Unlike previous work, the DRNN-SE employs GRUs to map user interaction histories to vectors in a Euclidean space shared with user and target item vectors. Experiments on three datasets demonstrated the DRNN-SE achieved similar MAP and NDCG scores to its dot-product cousin (DRNN-DP), and outperformed state-of-the-art tensor factorization and metric embedding algorithms by up to $\sim 50\%$. In addition, the use of FLANN to perform item recommendations improved query times by an order of magnitude over standard exact techniques. These results are particularly relevant for AUIs and in other domains (e.g., social robotics) where real-time personalized adaptations are often required.

As future work, the DRNN-SE can be easily extended to project side-information, such as user profiles and interaction element features into the latent space, e.g., via a convolutional neural network or a multilayer perceptron. Furthermore, the GRU can be substituted with alternative LSTM variants, which may result in improved performance for certain applications [16]. To reduce error variance, particularly with small training datasets (e.g., EqGraph), we can include prior information in the form of artificial (designer-provided) interaction traces, or provide uncertainty estimation via probabilistic approaches [9, 25]. Efficient incremental updates can enable better adaptation to distribution changes. Finally, we are in the process of integrating the DRNN with an adaptive user interface for network analysis [14] and expect to follow-up this paper with user studies.

REFERENCES

1. 2016. Tensorflow Candidate Sampling Algorithms Reference. (2016). **https://www.tensorflow.org/extras/candidate_sampling.pdf**
2. Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the Xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys '14*. 257–264.
3. John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI'14*, Vol. 461. 43–52.
4. Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *KDD '12*. 714–722.
5. Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*. 1724–1734.
6. Kim Drnec, Amar R. Marathe, Jamie R. Lukos, and Jason S. Metcalfe. 2016. From Trust in Automation to Decision Neuroscience. *Frontiers in Human Neuroscience* 10, August (2016).
7. Anis Elbahi and Mohamed Nazih Omri. 2015. Conditional Random Fields For Web User Task Recognition Based On Human Computer Interaction. In *WSCG*. 59–63.
8. Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized Ranking Metric Embedding for Next New POI Recommendation. In *IJCAI*. 2069–2075.
9. Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *ICML*.
10. Neil T Heffernan and Cristina Lindquist Heffernan. 2014. The ASSISTments ecosystem. *International Journal of Artificial Intelligence in Education* 24, 4 (2014), 470–497.
11. Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*. 1–9.
12. Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–80.
13. Eric Horvitz, Jack Breese, David Heckerman, David Hovel, Koos Rommelset, and Koos Rommelse. 1998. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *UAI'98*. 256–265.
14. Catherine Inibhunu, Scott Langevin, Scott Ralph, Nathan Kronefeld, Harold Soh, Greg A Jamieson, Scott Sanner, Sean W Kortschot, Chelsea Carrasco, and Madeleine White. 2016. Adapting level of detail in user interfaces for Cybersecurity operations. In *Resilience Week (RWS), 2016*. IEEE, 13–16.
15. Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *ACL'15*.
16. Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015a. An Empirical Exploration of Recurrent Network Architectures. In *ICML*, Vol. 37.
17. Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015b. An empirical exploration of recurrent network architectures. In *ICML*. 2342–2350.
18. Mohammad Khoshneshin and W Nick Street. 2010. Collaborative filtering via euclidean embedding. *ACM RecSys'10* (2010), 87–94.
19. Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*. 1–15.
20. Jiming Liu, Chi Kuen Wong, and Ka Keung Hui. 2003. An Adaptive User Interface Based on Personalized Learning. *IEEE Intelligent Systems* 18, 2 (2003), 52–57.
21. Marius Muja and David G. Lowe. 2014. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE PAMI* 36 (2014).
22. Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW '10*. 811.
23. Avi Rushinek and Sara F. Rushinek. 1986. What Makes Users Happy? *Commun. ACM* 29, 7 (July 1986), 594–598.
24. Chanop Silpa-Anan and Richard Hartley. 2008. Optimised KD-trees for fast image descriptor matching. In *CVPR*. 1–8.
25. Harold Soh. 2016. Distance-Preserving Probabilistic Embeddings with Side Information: Variational Bayesian Multidimensional Scaling Gaussian Process. In *IJCAI*. 2011–2017.
26. Brandon Taylor, Anind K. Dey, Daniel Siewiorek, and Asim Smailagic. 2016. Using Crowd Sourcing to Measure the Effects of System Response Delays on User Engagement. In *CHI '16*. 4413–4422.
27. Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized next-song recommendation in online karaokes. *RecSys '13* (2013), 137–140.
28. Vasilios Zarikas. 2007. Modeling decisions under uncertainty in adaptive user interfaces. *Universal Access in the Information Society* 6, 1 (2007), 87–101.