Context Uncertainty in Contextual Bandits with Applications to Recommender Systems

Hao Wang[1], **Yifei Ma**[2], **Hao Ding**[2], **Yuyang Wang**[2]

[1]Department of Computer Science, Rutgers University [2]AWS AI Lab
hw488@cs.rutgers.edu, {yifeim,haodin,yuyawang}@amazon.com

Abstract

Recurrent neural networks have proven effective in modeling sequential user feedbacks for recommender systems. However, they usually focus solely on item relevance and fail to effectively explore diverse items for users, therefore harming the system performance in the long run. To address this problem, we propose a new type of recurrent neural networks, dubbed recurrent exploration networks (REN), to jointly perform representation learning and effective exploration in the latent space. REN tries to balance relevance and exploration while taking into account the uncertainty in the representations. Our theoretical analysis shows that REN can preserve the rate-optimal sublinear regret even when there exists uncertainty in the learned representations. Our empirical study demonstrates that REN can achieve satisfactory long-term rewards on both synthetic and real-world recommendation datasets, outperforming state-of-the-art models.

Introduction

Modeling and predicting sequential user feedbacks is a core problem in modern e-commerce recommender systems. In this regard, recurrent neural networks (RNN) have shown great promise since they can naturally handle sequential data [15, 30, 5, 26]. While these RNN-based models can effectively learn representations in the latent space to achieve satisfactory immediate recommendation accuracy, they typically focus solely on relevance and fall short of effective exploration in the latent space, leading to poor performance in the long run. For example, a recommender system may keep recommending action movies to a user once it learns that she likes such movies. This may increase immediate rewards, but the lack of exploration in other movie genres can certainly be detrimental to long-term rewards.

So, how does one effectively explore diverse items for users while retaining the representation power offered by RNN-based recommenders. We note that the learned representations in the latent space are crucial for these models' success. Therefore we propose recurrent exploration networks (REN) to explore diverse items in the latent space learned by RNN-based models. REN tries to balance relevance and exploration during recommendations using the learned representations.

One roadblock is that effective exploration relies heavily on well learned representations, which in turn require sufficient exploration; this is a chicken-and-egg problem. In a case where RNN learns unreasonable representations (e.g., all items have the same representations), exploration in the latent space is meaningless. To address this problem, we enable REN to take into account the uncertainty of the learned representations as well during recommendations. Essentially items whose representations have higher uncertainty can be explored more often. Such a model can be seen as a contextual bandit algorithm that is aware of the uncertainty for each context. Our contributions are as follows:

1. We propose REN as a new type of RNN to balance relevance and exploration during recommendation, yielding satisfactory long-term rewards.
2. Our theoretical analysis shows that there is an upper confidence bound related to uncertainty in learned representations. With such a bound implemented in the algorithm, REN can achieve the same rate-optimal sublinear regret. To the best of our knowledge, we are the first to study the regret bounds under "context uncertainty".
3. Experiments of joint learning and exploration on both synthetic and real-world temporal datasets show that REN significantly improve long-term rewards over state-of-the-art RNN-based recommenders.

Related Work

Deep Learning for Recommender Systems. Deep learning (DL) has been playing a key role in modern recommender systems [31, 35, 40, 38, 39, 24, 7, 10, 33, 13]. [31] uses restricted Boltzmann machine to perform collaborative filtering in recommender systems. [40], [39], and [24] devise Bayesian deep learning models [41, 42, 37] to significantly improve recommendation performance. In terms of sequential (or session-based) recommender systems [15, 30, 4, 21, 25, 44, 26], GRU4Rec [15] was first proposed to use gated recurrent units (GRU) [8], an RNN variant with gating mechanism, for recommendation. Since then, follow-up works such as hierarchical GRU [30], temporal convolutional networks (TCN) [4], and hierarchical RNN (HRNN) [26] have tried to achieve improvement in accuracy with the help of cross-session information [30], causal convolutions [4], as well as control signals [26]. We note that our REN does not assume spe-

cific RNN architectures (e.g., GRU or TCN) and is therefore *compatible with different RNN-based (or more generally DL-based) models*, as shown in later sections.

Contextual Bandits. Contextual bandit algorithms such as LinUCB [22] and its variants [45, 1, 23, 20, 11, 19, 27, 46] have been proposed to tackle the exploitation-exploration trade-off in recommender systems and successfully improve upon context-free bandit algorithms [3]. Similar to [3], theoretical analysis shows that LinUCB variants could achieve a rate-optimal regret bound [9]. However, these methods either assume observed context [46] or are incompatible with neural networks [23, 45]. In contrast, REN as a contextual bandit algorithm runs in the latent space and assumes user models based on RNN; therefore it is compatible with state-of-the-art RNN-based recommender systems.

Diversity-Inducing Models. Various works have focused on inducing diversity in recommender systems [28, 2, 43, 6]. Usually such a system consists of a submodular function, which measures the diversity among items, and a relevance prediction model, which predicts relevance between users and items. Examples of submodular functions include the probabilistic coverage function [16] and facility location diversity (FILD) [34], while relevance prediction models can be Gaussian processes [36], linear regression [45], etc. These models typically focus on improving *diversity among recommended items in a slate* at the cost of accuracy. In contrast, REN's goal is to optimize for long-term rewards through improving *diversity between previous and recommended items*. We include some slate generation in our real-data experiments for completeness.

Recurrent Exploration Networks

In this section we first describe the general notations and how RNN can be used for recommendation, briefly review determinantal point processes (DPP) as a diversity-inducing model as well as their connection to exploration in contextual bandits, and then introduce our proposed REN framework.

Notation and RNN-Based Recommender Systems

Notation. We consider the problem of sequential recommendations where the goal is to predict the item a user interacts with (e.g., click or purchase) at time t, denoted as $\mathbf{e}_{k_t}$, given her previous interaction history $\mathbf{E}_t = [\mathbf{e}_{k_\tau}]_{\tau=1}^{t-1}$. Here k_t is the index for the item at time t, $\mathbf{e}_{k_t} \in \{0,1\}^K$ is a one-hot vector indicating an item, and K is the number of total items. We denote the item embedding (encoding) for $\mathbf{e}_{k_t}$ as $\mathbf{x}_{k_t} = f_e(\mathbf{e}_{k_t})$, where $f_e(\cdot)$ is the encoder as a part of the RNN. Correspondingly we have $\mathbf{X}_t = [\mathbf{x}_{k_\tau}]_{\tau=1}^{t-1}$. Strictly speaking, in an online setting where the model updates at every time step t, $\mathbf{x}_k$ also changes over time; in Sec. we use $\mathbf{x}_k$ as a shorthand for $\mathbf{x}_{t,k}$ for simplicity. We use $\|\mathbf{z}\|_\infty = \max_i |\mathbf{z}^{(i)}|$ to denote the L_∞ norm, where the superscript (i) means the i-th entry of the vector $\mathbf{z}$.

RNN-Based Recommender Systems. Given the interaction history $\mathbf{E}_t$, the RNN generates the user embedding at time t as $\boldsymbol{\theta}_t = R([\mathbf{x}_{k_\tau}]_{\tau=1}^{t-1})$, where $\mathbf{x}_{k_\tau} = f_e(\mathbf{e}_{k_\tau}) \in \mathbb{R}^d$, and $R(\cdot)$ is the recurrent part of the RNN. Assuming tied weights, the score for each candidate item is then computed as $p_{k,t} = \mathbf{x}_k^\top \boldsymbol{\theta}_t$. As the last step, the recommender system will recommend the items with the highest scores to the user. Note that the subscript k indexes the items, and is equivalent to an 'action', usually denoted as a, in the context of bandit algorithms.

Determinantal Point Processes for Diversity and Exploration

Determinantal point processes (DPP) consider an item selection problem where each item is represented by a feature vector $\mathbf{x}_t$. Diversity is achieved by picking a subset of items to cover the maximum volume spanned by the items, measured by the log-determinant of the corresponding kernel matrix, $\ker(\mathbf{X}_t) = \log\det(\mathbf{I}_K + \mathbf{X}_t\mathbf{X}_t^\top)$, where $\mathbf{I}_K$ is included to prevent singularity. Intuitively, DPP penalizes colinearity, which is an indicator that the topics of one item are already covered by the other topics in the full set. The log-determinant of a kernel matrix is also a submodular function [12], which implies a $(1-1/e)$-optimal guarantees from greedy solutions. The greedy algorithm for DPP via the matrix determinant lemma is

$$\begin{aligned} &\operatorname{argmax}_k \ \log\det(\mathbf{I}_d + \mathbf{X}_t^\top\mathbf{X}_t + \mathbf{x}_k\mathbf{x}_k^\top) \qquad (1)\\ &\qquad\qquad - \log\det(\mathbf{I}_d + \mathbf{X}_t^\top\mathbf{X}_t)\\ &= \operatorname{argmax}_k \ \log(1 + \mathbf{x}_k^\top(\mathbf{I}_d + \mathbf{X}_t^\top\mathbf{X}_t)^{-1}\mathbf{x}_k) \qquad (2)\\ &= \operatorname{argmax}_k \ \sqrt{\mathbf{x}_k^\top(\mathbf{I}_d + \mathbf{X}_t^\top\mathbf{X}_t)^{-1}\mathbf{x}_k}. \qquad (3) \end{aligned}$$

Interestingly, note that $\sqrt{\mathbf{x}_k^\top(\mathbf{I}_d + \mathbf{X}_t^\top\mathbf{X}_t)^{-1}\mathbf{x}_k}$ has the same form as the confidence interval in LinUCB [22], a commonly used contextual bandit algorithm to boost exploration and achieve long-term rewards, suggesting a connection between diversity and long-term rewards [45]. Intuitively, this makes sense in recommender systems since encouraging diversity relative to user history (*as well as diversity in a slate of recommendations in our experiments*) naturally explores user interest previously unknown to the model, leading to much higher long-term rewards, as shown in Sec. 12.

Recurrent Exploration Networks

Exploration Term. Based on the intuition above, we can modify the user-item score $p_{k,t} = \mathbf{x}_k^\top\boldsymbol{\theta}_t$ to include a diversity (exploration) term, leading to the new score

$$p_{k,t} = \mathbf{x}_k^\top\boldsymbol{\theta}_t + \lambda_d\sqrt{\mathbf{x}_k^\top(\mathbf{I}_d + \mathbf{X}_t^\top\mathbf{X}_t)^{-1}\mathbf{x}_k}, \qquad (4)$$

where the first term is the relevance score and the second term is the exploration score (measuring diversity *between previous and recommended items*). $\boldsymbol{\theta}_t = R(\mathbf{X}_t) = R([\mathbf{x}_{k_\tau}]_{\tau=1}^{t-1})$ is RNN's hidden states at time t representing the user embedding. The hyperparameter λ_d aims to balance two terms.

Uncertainty Term for Context Uncertainty. At first blush, given the user history the system using Eqn. 4 will recommend items that are (1) relevant to the user's interest and (2) diverse from the user's previous items. However, this only works when item embeddings $\mathbf{x}_k$ are correctly learned. Unfortunately, the quality of learned item embeddings, in turn, relies heavily on the effectiveness of exploration, leading to

Algorithm 1: Recurrent Exploration Networks (REN)

Input: λ_d, λ_u, initialized REN model with the encoder, i.e., $R(\cdot)$ and $f_e(\cdot)$.
for $t = 1, 2, \ldots, T$ **do**
  Obtain item embeddings from REN:
    $\boldsymbol{\mu}_{k_\tau} \leftarrow f_e(\mathbf{e}_{k_\tau})$ for all $\tau \in \{1, 2, \ldots, t-1\}$.
  Obtain the current user embedding from REN:
    $\boldsymbol{\theta}_t \leftarrow R(\mathbf{D}_t)$.
  Compute $\mathbf{A}_t \leftarrow \mathbf{I}_d + \sum_{\tau \in \Psi_t} \boldsymbol{\mu}_{k_\tau}^\top \boldsymbol{\mu}_{k_\tau}$.
  Obtain candidate items' embeddings from REN:
    $\boldsymbol{\mu}_k \leftarrow f_e(\mathbf{e}_k)$, where $k \in [K]$.
  Obtain candidate items' uncertainty estimates $\boldsymbol{\sigma}_k$, where $k \in [K]$.
  for $k \in [K]$ **do**
    Obtain the score for item k at time t:
      $p_{k,t} \leftarrow \boldsymbol{\mu}_k^\top \boldsymbol{\theta}_t + \lambda_d \sqrt{\boldsymbol{\mu}_k^\top \mathbf{A}_t^{-1} \boldsymbol{\mu}_k} + \lambda_u \|\boldsymbol{\sigma}_k\|_\infty$.
  end
  Recommend item $k_t \leftarrow \text{argmax}_k\, p_{t,k}$ and collect user feedbacks.
  Update the REN model $R(\cdot)$ and $f_e(\cdot)$ using collected user feedbacks.
end

a chicken-and-egg problem. To address this problem, one also needs to consider the uncertainty of the learned item embeddings. Assuming the item embedding $\mathbf{x}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$, where $\boldsymbol{\Sigma}_k = \mathbf{diag}(\boldsymbol{\sigma}_k^2)$, we have the final score for REN:

$$p_{k,t} = \boldsymbol{\mu}_k^\top \boldsymbol{\theta}_t + \lambda_d \sqrt{\boldsymbol{\mu}_k^\top (\mathbf{I}_d + \mathbf{D}_t^\top \mathbf{D}_t)^{-1} \boldsymbol{\mu}_k} + \lambda_u \|\boldsymbol{\sigma}_k\|_\infty, \tag{5}$$

where $\boldsymbol{\theta}_t = R(\mathbf{D}_t) = R([\boldsymbol{\mu}_{k_\tau}]_{\tau=1}^{t-1})$ and $\mathbf{D}_t = [\boldsymbol{\mu}_{k_\tau}]_{\tau=1}^{t-1}$. The term $\boldsymbol{\sigma}_k$ quantifies the uncertainty for each dimension of $\mathbf{x}_k$, meaning that items whose embeddings REN is uncertain about are more likely to be recommended. Therefore with the third term, REN can naturally balance among relevance, diversity (*relative to user history*), and uncertainty during exploration.

Putting It All Together. Algorithm 1 shows the overview of REN. Note that the difference between REN and traditional RNN-based recommenders is only in the inference stage. During training (Line 16 of Algorithm 1), one can train REN only with the relevance term using models such as GRU4Rec and HRNN. In the experiments, we use uncertainty estimates $\mathbf{diag}(\boldsymbol{\sigma}_k) = 1/\sqrt{n_k}\ \mathbf{I}_d$, where n_k is item k's total number of impressions (i.e., the number of times item k has been recommended) for all users. The intuition is that: the more frequently item k is recommended, the more frequently its embedding $\mathbf{x}_k$ gets updated, the faster $\boldsymbol{\sigma}_k$ decreases.[1] Our preliminary experiments show that $1/\sqrt{n_k}$ does decrease at the rate of $O(1/\sqrt{t})$, meaning that the assumption in Lemma 4 is satisfied. From the Bayesian perspective, $1/\sqrt{n_k}$ may not accurately reflect the uncertainty of the learned x_k, which is a limitation of our model. In principle, one can learn $\boldsymbol{\sigma}_k$ from data using the reparameterization trick [18] with a Gaussian prior on x_k and examine whether $\boldsymbol{\sigma}_k$ the assumption in Lemma 4; this would be interesting future work.

Linearity in REN. REN only needs a linear bandit model; REN's output $\mathbf{x}_k^\top \boldsymbol{\theta}_t$ is linear w.r.t. $\boldsymbol{\theta}$ and $\mathbf{x}_k$. Note that NeuralUCB [46] is a powerful nonlinear extension of LinUCB, i.e., its output is nonlinear w.r.t. $\boldsymbol{\theta}$ and $\mathbf{x}_k$. Extending REN's output from $\mathbf{x}_k^\top \boldsymbol{\theta}_t$ to a nonlinear function $f(\mathbf{x}_k, \boldsymbol{\theta}_t)$ as in NeuralUCB is also interesting future work.[2]

Beyond RNN. Note that our methods and theory go beyond RNN-based models and can be naturally extended to any latent factor models including transformers, MLPs, and matrix factorization. The key is the user embedding $\boldsymbol{\theta}_t = R(\mathbf{X}_t)$, which can be instantiated with an RNN, a transformer, or a matrix-factorization model.

Theoretical Analysis

With REN's connection to contextual bandits, we can prove that with proper λ_d and λ_u, Eqn. 5 is actually the upper confidence bound that leads to long-term rewards with a rate-optimal regret bound.

Reward Uncertainty versus Context Uncertainty. Note that unlike existing works which primarily consider the randomness from the reward, we take into consideration the uncertainty resulted from the context, i.e., *context uncertainty*. More specifically, existing works assume deterministic $\mathbf{x}$ and only assume randomness in the reward, i.e., they assume that $r = \mathbf{x}^\top \boldsymbol{\theta} + \epsilon$, and therefore r's randomness is independent of $\mathbf{x}$. The problem with this formulation is that they assume $\mathbf{x}$ is deterministic and therefore the model only has a point estimate of the item embedding $\mathbf{x}$, but does not have uncertainty estimation for such $\mathbf{x}$. We find that such uncertainty estimation is crucial for exploration; if the model is uncertain about $\mathbf{x}$, it can then explore more on the corresponding item.

To facilitate analysis, we follow common practice [3, 9] to divide the procedure of REN into "BaseREN" (Algorithm 2) and "SupREN" stages correspondingly. Essentially SupREN introduces $S = \ln T$ levels of elimination (with s as an index) to filter out low-quality items and ensures that the assumption holds (see the Supplement for details of SupREN).

In this section, we first provide a high probability bound for BaseREN with uncertain embeddings (context), and derive an upper bound for the regret. As mentioned in Sec. , for the online setting where the model updates at every time step t, $\mathbf{x}_k$ also changes over time. Therefore in this section we use

[1] There are some caveats in general. $\mathbf{diag}(\boldsymbol{\sigma}_k) \propto \mathbf{I}_d$ assumes that all coordinates of x shrink at the same rate. However, REN exploration mechanism associates n_k with the total variance of the features of an item. This may not ensure all feature dimensions to be equally explored. See [17] for a different algorithm that analyzes the exploration of the low-rank feature space.

[2] In other words, we did not fully explain why x could be shared between non-linear RNN and the uncertainty bounds based on linear models. On the other hand, we did observe promising empirical results, which may encourage interested readers to dive deep into different theoretical analyses.

Algorithm 2: BaseREN: Basic REN Inference at Step t

Input: α, $\Psi_t \subseteq \{1, 2, \ldots, t-1\}$.
Obtain item embeddings from REN: $\boldsymbol{\mu}_{\tau,k_\tau} \leftarrow f_e(\mathbf{e}_{\tau,k_\tau})$ for all $\tau \in \Psi_t$.
Obtain user embedding: $\boldsymbol{\theta}_t \leftarrow R(\mathbf{D}_t)$.
$\mathbf{A}_t \leftarrow \mathbf{I}_d + \sum_{\tau\in\Psi_t} \boldsymbol{\mu}_{\tau,k_\tau}^\top \boldsymbol{\mu}_{\tau,k_\tau}$.
Obtain candidate items' embeddings: $\boldsymbol{\mu}_{t,k} \leftarrow f_e(\mathbf{e}_{t,k})$, where $k \in [K]$.
Obtain candidate items' uncertainty estimates $\boldsymbol{\sigma}_{t,k}$, where $k \in [K]$.
for $a \in [K]$ **do**
  $s_{t,k} = \sqrt{\boldsymbol{\mu}_{t,k}^\top \mathbf{A}_t^{-1} \boldsymbol{\mu}_{t,k}}$
  $w_{t,k} \leftarrow (\alpha+1)s_{t,k} + (4\sqrt{d} + 2\sqrt{\ln \frac{TK}{\delta}})\|\boldsymbol{\sigma}_{t,k}\|_\infty$.
  $\widehat{r}_{t,k} \leftarrow \boldsymbol{\theta}_t^\top \boldsymbol{\mu}_{t,k}$.
end
Recommend item $k \leftarrow \operatorname{argmax}_k \widehat{r}_{t,k} + w_{t,k}$.

$\mathbf{x}_{t,k}$, $\boldsymbol{\mu}_{t,k}$, $\boldsymbol{\Sigma}_{t,k}$, and $\boldsymbol{\sigma}_{t,k}$ in place of $\mathbf{x}_k$, $\boldsymbol{\mu}_k$, $\boldsymbol{\Sigma}_k$, and $\boldsymbol{\sigma}_k$ from Sec. to be rigorous.

Assumption 1. *Assume there exists an optimal $\boldsymbol{\theta}^*$, with $\|\boldsymbol{\theta}^*\| \le 1$, and $\mathbf{x}_{t,k}^*$ such that $\mathbf{E}[r_{t,k}] = {\mathbf{x}_{t,k}^*}^\top \boldsymbol{\theta}^*$. Further assume that there is an effective distribution $\mathcal{N}(\boldsymbol{\mu}_{t,k}, \boldsymbol{\Sigma}_{t,k})$ such that $\mathbf{x}_{t,k}^* \sim \mathcal{N}(\boldsymbol{\mu}_{t,k}, \boldsymbol{\Sigma}_{t,k})$ where $\boldsymbol{\Sigma}_{t,k} = \boldsymbol{diag}(\boldsymbol{\sigma}_{t,k}^2)$. Thus, the true underlying context is unavailable, but we are aided with the knowledge that it is generated by a multivariate normal with known parameters*[3].

Upper Confidence Bound for Uncertain Embeddings

For simplicity denote the item embedding (context) as $\mathbf{x}_{t,k}$, where t indexes the rounds (time steps) and k indexes the items. We define:

$$s_{t,k} = \sqrt{\boldsymbol{\mu}_{t,k}^\top \mathbf{A}_t^{-1} \boldsymbol{\mu}_{t,k}} \in \mathbb{R}_+, \quad \mathbf{D}_t = [\boldsymbol{\mu}_{\tau,k_\tau}]_{\tau\in\Psi_t} \in \mathbb{R}^{|\Psi_t|\times d},$$
$$\mathbf{y}_t = [r_{\tau,k_\tau}]_{\tau\in\Psi_t} \in \mathbb{R}^{|\Psi_t|\times 1}, \quad \mathbf{A}_t = \mathbf{I}_d + \mathbf{D}_t^\top \mathbf{D}_t,$$
$$\mathbf{b}_t = \mathbf{D}_t^\top \mathbf{y}_t, \qquad \widehat{r}_{t,k} = \boldsymbol{\mu}_{t,k}^\top \hat{\boldsymbol{\theta}}_t = \boldsymbol{\mu}_{t,k}^\top \mathbf{A}_t^{-1} \mathbf{b}_t, \tag{6}$$

where $\mathbf{y}_t$ is the collected user feedback. Lemma 1 below shows that with $\lambda_d = 1 + \alpha = 1 + \sqrt{\frac{1}{2}\ln\frac{2TK}{\delta}}$ and $\lambda_u = 4\sqrt{d} + 2\sqrt{\ln\frac{TK}{\delta}}$, Eqn. 5 is the upper confidence bound with high probability, meaning that Eqn. 5 upper bounds the true reward with high probability, which makes it a reasonable score for recommendations.

Lemma 1 (Confidence Bound). *With probability at least $1 - 2\delta/T$, we have for all $k \in [K]$ that*

$$|\widehat{r}_{t,k} - {\mathbf{x}_{t,k}^*}^\top \boldsymbol{\theta}^*| \le (\alpha+1)s_{t,k} + (4\sqrt{d} + 2\sqrt{\ln\frac{TK}{\delta}})\|\boldsymbol{\sigma}_{t,k}\|_\infty,$$

where $\|\boldsymbol{\sigma}_{t,k}\|_\infty = \max_i |\boldsymbol{\sigma}_{t,k}^{(i)}|$ is the L_∞ norm.

The proof is in the Supplement. This upper confidence bound above provides important insight on why Eqn. 5 is reasonable as a final score to select items in Algorithm 1 as well as the choice of hyperparameters λ_d and λ_u.

RNN to Estimate $\boldsymbol{\theta}_t$. REN uses RNN to approximate $\mathbf{A}_t^{-1}\mathbf{b}_t$ (useful in the proof of Lemma 1) in Eqn. 6. Note that a linear RNN with tied weights and a single time step is equivalent to linear regression (LR); therefore RNN is a more general model to estimate $\boldsymbol{\theta}_t$. Compared to LR, RNN-based recommenders can naturally incorporate new user history by incrementally updating the hidden states ($\boldsymbol{\theta}_t$ in REN), without the need to solve a linear equation. Interestingly, one can also see RNN's recurrent computation as a simulation (approximation) for solving equations via iterative updating.

Regret Bound

Lemma 1 above provides an estimate of the reward's upper bound at time t. Based on this estimate, one natural next step is to analyze the regret after all T rounds. Formally, we define the regret of the algorithm after T rounds as

$$B(T) = \sum_{t=1}^{T} r_{t,k_t^*} - \sum_{t=1}^{T} r_{t,k_t}, \tag{7}$$

where k_t^* is the optimal item (action) k at round t that maximizes $\mathbf{E}[r_{t,k}] = {\mathbf{x}_{t,k}^*}^\top \boldsymbol{\theta}^*$, and k_t is the action chose by the algorithm at round t. Similar to [3], SupREN calls BaseREN as a sub-routine. In this subsection, we derive the regret bound for SupREN with uncertain item embeddings.

Lemma 2. *With probability $1 - 2\delta S$, for any $t \in [T]$ and any $s \in [S]$, we have: (1) $|\widehat{r}_{t,k} - \mathbf{E}[r_{t,k}]| \le w_{t,k}$ for any $k \in [K]$, (2) $k_t^* \in \hat{A}_s$, and (3) $\mathbf{E}[r_{t,k_t^*}] - \mathbf{E}[r_{t,k}] \le 2^{(3-s)}$ for any $k \in \hat{A}_s$.*

Lemma 3. *In BaseREN, we have: $(1+\alpha)\sum_{t\in\Psi_{T+1}} s_{t,k_t} \le 5\cdot(1+\alpha^2)\sqrt{d|\Psi_{T+1}|}$.*

Lemma 4. *Assuming $\|\boldsymbol{\sigma}_{1,k}\|_\infty = 1$ and $\|\boldsymbol{\sigma}_{t,k}\|_\infty \le \frac{1}{\sqrt{t}}$ for any k and t, then for any k, we have the upper bound: $\sum_{t\in\Psi_{T+1}} \|\boldsymbol{\sigma}_{t,k}\|_\infty \le \sqrt{|\Psi_{T+1}|}$.*

Essentially Lemma 2 links the regret $B(T)$ to the width of the confidence bound $w_{t,k}$ (Line 9 of Algorithm 2 or the last two terms of Eqn. 5). Lemma 3 and Lemma 4 then connect $w_{t,k}$ to $\sqrt{|\Psi_{T+1}|} \le \sqrt{T}$, which is sublinear in T; this is the key to achieve a sublinear regret bound. Note that $\hat{A}_s$ is defined inside Algorithm 2 (SupREN) of the Supplement.

Interestingly, Lemma 4 states that the uncertainty only needs to decrease at the rate $\frac{1}{\sqrt{t}}$, which is consistent with our choice of $\mathbf{diag}(\boldsymbol{\sigma}_k) = 1/\sqrt{n_k}\,\mathbf{I}_d$ in Sec. , where n_k is item

[3] Here we omit the identifiability issue of $\mathbf{x}_{t,k}^*$ and assume that there is a unique $\mathbf{x}_{t,k}^*$ for clarity.

k's total number of impressions for all users. As the last step, Lemma 5 and Theorem 1 below build on all lemmas above to derive the final sublinear regret bound.

Lemma 5. *For all $s \in [S]$,*

$$|\Psi_{T+1}^{(s)}| \leq 2^s \cdot (5(1+\alpha^2)\sqrt{d|\Psi_{T+1}^{(s)}|} + 4\sqrt{dT} + 2\sqrt{T \ln \frac{TK}{\delta}}).$$

Theorem 1. *If SupREN is run with $\alpha = \sqrt{\frac{1}{2} \ln \frac{2TK}{\delta}}$, with probability at least $1-\delta$, the regret of the algorithm is*

$$\begin{aligned} B(T) &\leq 2\sqrt{T} + 92 \cdot (1 + \ln \frac{2TK(2\ln T + 2)}{\delta})^{\frac{3}{2}} \sqrt{Td} \\ &= O(\sqrt{Td \ln^3(\frac{KT \ln(T)}{\delta})}), \end{aligned}$$

The full proofs of all lemmas and the theorem are in the Supplement. Theorem 1 shows that even with the uncertainty in the item embeddings (i.e., context uncertainty), our proposed REN can achieve the same rate-optimal sublinear regret bound.

Experiments

In this section, we evaluate our proposed REN on both synthetic and real-world datasets.

Experiment Setup and Compared Methods

Joint Learning and Exploration Procedure in Temporal Data. To effectively verify REN's capability to boost long-term rewards, we adopt an online experiment setting where data is divided into different time intervals $[T_0, T_1), [T_1, T_2), \ldots, [T_{M-1}, T_M]$. RNN (including REN and its baselines) is then trained and evaluated in a rolling manner: (1) RNN is trained using data in $[T_0, T_1)$; (2) RNN is evaluated using data in $[T_1, T_2)$ and collects feedbacks (rewards) for its recommendations; (3) RNN uses newly collected feedbacks from $[T_1, T_2)$ to finetune the model; (4) Repeat the previous two steps using data from the next time interval. Note that different from traditional offline and one-step evaluation, corresponding to only Step (1) and (2), our setting performs joint learning and exploration in temporal data, and therefore is more realistic and closer to production systems.

Long-Term Rewards. Since the goal is to evaluate long-term rewards, we are mostly interested in the rewards during the last (few) time intervals. Conventional RNN-based recommenders do not perform exploration and are therefore much easier to saturate at a relatively low reward. In contrast, REN with its effective exploration can achieve nearly optimal rewards in the end.

Compared Methods. We compare REN variants with state-of-the-art RNN-based recommenders including **GRU4Rec** [15], **TCN** [4], **HRNN** [26]. Since REN can use any RNN-based recommenders as a base model, we evaluate three REN variants in the experiments: **REN-G**, **REN-T**, and **REN-H**, which use GRU4Rec, TCN, and HRNN as base models, respectively. Additionally we also evaluate **REN-1,2**, an REN variant without the third term of Eqn. 5, and **REN-1,3**, one without the second term of Eqn. 5, as an ablation study. Both REN-1,2 and REN-1,3 use GRU4Rec as the base model. As references we also include **Oracle**, which always achieves optimal rewards, and **Random**, which randomly recommends one item from the full set. For REN variants we choose λ_d from $\{0.001, 0.005, 0.01, 0.05, 0.1\}$ and set $\lambda_u = \sqrt{10}\lambda_d$. Other hyperparameters in the RNN base models are kept the same for fair comparison (see the Supplement for more details on neural network architectures, hyperparameters, and their sensitivity analysis).

Connection to Reinforcement Learning (RL) and Bandits. REN-1,2 (in Fig. 2) can be seen as a simplified version of 'randomized least-squares value iteration' (an RL approach proposed in [29]) or an adapted version of contextual bandits, while REN-1,3 (in Fig. 2) is an advanced version of ϵ-greedy exploration in RL. Note that REN is orthogonal to RL [32] and bandit methods.

Simulated Experiments

Datasets. Following the setting described in Sec. 12, we start with three synthetic datasets, namely *SYN-S*, *SYN-M*, and *SYN-L*, which allow complete control on the simulated environments. We assume 8-dimensional latent vectors, which are unknown to the models, for each user and item, and use the inner product between user and item latent vectors as the reward. Specifically, for each latent user vector $\boldsymbol{\theta}^*$, we randomly choose 3 entries to set to $1/\sqrt{3}$ and set the rest to 0, keeping $\|\boldsymbol{\theta}^*\|_2 = 1$. We generate $C_2^8 = 28$ unique item latent vectors. Each item latent vector $\mathbf{x}_k^*$ has 2 entries set to $1/\sqrt{2}$ and the other 6 entries set to 0 so that $\|\mathbf{x}_k^*\|_2 = 1$.

We assume 15 users in our datasets. *SYN-S* contains exactly 28 items, while *SYN-M* repeats each unique item latent vector for 10 times, yielding 280 items in total. Similarly, *SYN-L* repeats for 50 times, therefore yielding 1400 items in total. The purpose of allowing different items to have identical latent vectors is to investigate REN's capability to explore in the compact latent space rather than the large item space. All users have a history length of 60.

Simulated Environments. With the generated latent vectors, the simulated environment runs as follows: At each time step t, the environment randomly chooses one user and feed the user's interaction history $\mathbf{X}_t$ (or $\mathbf{D}_t$) into the RNN recommender. The recommender then recommends the top 4 items to the user. The user will select the item with the highest ground-truth reward ${\boldsymbol{\theta}^*}^\top \mathbf{x}_k^*$, after which the recommender will collect the selected item with the reward and finetune the model.

Results. Fig. 1 shows the rewards over time for different methods. Results are averaged over 3 runs and we plot the rolling average with a window size of 100 to prevent clutter. As expected, conventional RNN-based recommenders saturate at around the 500-th time step, while all REN variants successfully achieve nearly optimal rewards in the end. One interesting observation is that REN variants obtain rewards lower than the "Random" baseline at the beginning, meaning that they are sacrificing immediate rewards to perform exploration in exchange for long-term rewards.

Ablation Study. Fig. 2 shows the rewards over time for

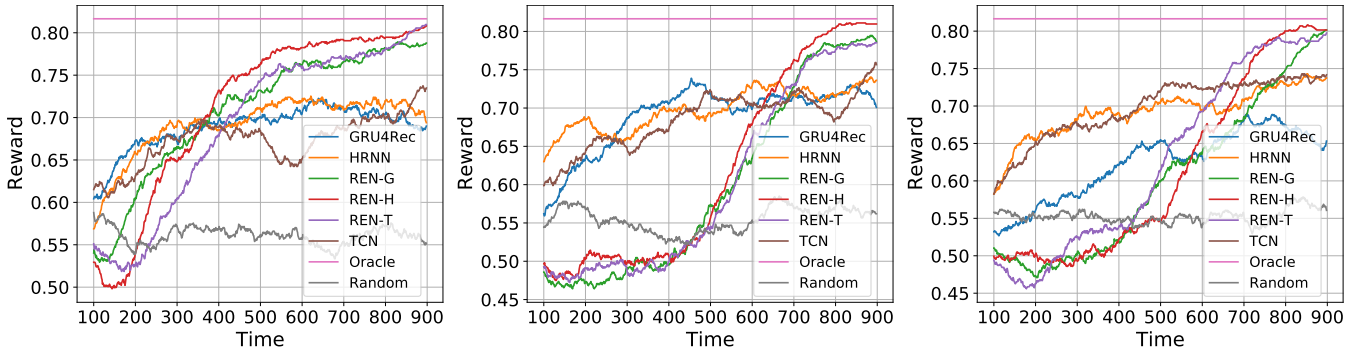

Figure 1: Results for different methods in *SYN-S* (**left** with 28 items), *SYN-M* (**middle** with 280 items), and *SYN-L* (**right** with 1400 items). One time step represents one interaction step, where in each interaction step the model recommends 3 items to the user and the user interacts with one of them. In all cases, REN models with diversity-based exploration lead to final convergence, whereas models without exploration get stuck at local optima.

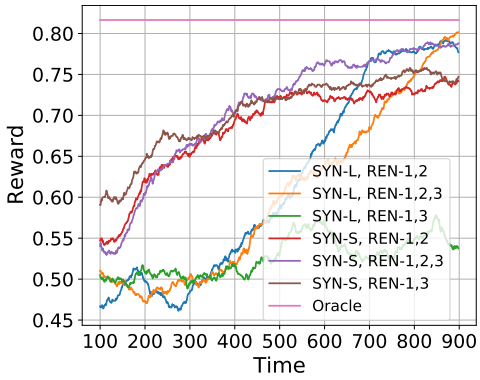

Figure 2: Ablation study on different terms of REN. 'REN-1,2,3' refers to the full 'REN-G' model.

REN-G (i.e., REN-1,2,3), REN-1,2, and REN-1,3 in *SYN-S* and *SYN-L*. We observe that REN-1,2, with only the relevance (first) and diversity (second) terms of Eqn. 5, saturates prematurely in *SYN-S*. On the other hand, the reward of REN-1,3, with only the relevance (first) and uncertainty (third) term, barely increases over time in *SYN-L*. In contrast, the full REN-G works in both *SYN-S* and *SYN-L*. This is because without the uncertainty term, REN-1,2 fails to effectively choose items with uncertain embeddings to explore. REN-1,3 ignores the diversity in the latent space and tends to explore items that have rarely been recommended; such exploration directly in the item space only works when the item number is small, e.g., in *SYN-S*.

Hyperparameters. For the base models GRU4Rec, TCN, and HRNN, we use identical network architectures and hyperparemeters whenever possible following [15, 4, 26]. Each RNN consists of an encoding layer, a core RNN layer, and a decoding layer. We set the number of hidden neurons to 32 for all models including REN variants. Fig. 1 in the Supplement shows the REN-G's performance for different λ_d (note that we fix $\lambda_u = \sqrt{10}\lambda_d$) in *SYN-S*, *SYN-M*, and *SYN-L*. We can observe stable REN performance across a wide range of λ_d. As expected, REN-G's performance is closer to GRU4Rec when λ_d is small.

Real-World Experiments

***MovieLens-1M*.** We use *MovieLens-1M* [14] containing 3,900 movies and 6,040 users with an experiment setting similar to Sec. 12. Each user has 120 interactions, and we follow the joint learning and exploration procedure described in Sec. 12 to evaluate all methods (more details in the Supplement). All models recommend 10 items at each round for a chosen user, and the precision@10 is used as the reward. Fig. 3(left) shows the rewards over time averaged over all 6,040 users. As expected, REN variants with different base models are able to achieve higher long-term rewards compared to their non-REN counterparts.

***Trivago*.** We also evaluate the proposed methods on *Trivago*[4], a hotel recommendation dataset with 730,803 users, 926,457 items, and 910,683 interactions. We use a subset with 57,778 users, 387,348 items, and 108,713 interactions and slice the data into $M = 48$ one-hour time intervals for the online experiment (see the Supplement for details on data pre-processing). Different from *MovieLens-1M*, *Triavago* has impression data available: at each time step, besides which item is clicked by the user, we also know which 25 items are being shown to the user. Such information makes the online evaluation more realistic, as we now know the ground-truth feedback if an arbitrary subset of the 25 items are presented to the user. At each time step of the online experiments, all methods will choose 10 items from the 25 items to recommend the current user and collect the feedback for these 10 items as data for finetuning. We pretrain the model using *all* 25 *items* from the first 13 hours before starting the online evaluation. Fig. 3(middle) shows the mean reciprocal rank (MRR), the official metric used in the RecSys Challenge, for different methods. As expected, the baseline RNN (e.g., GRU4Rec)

[4]More details are available at https://recsys.trivago.cloud/challenge/dataset/.

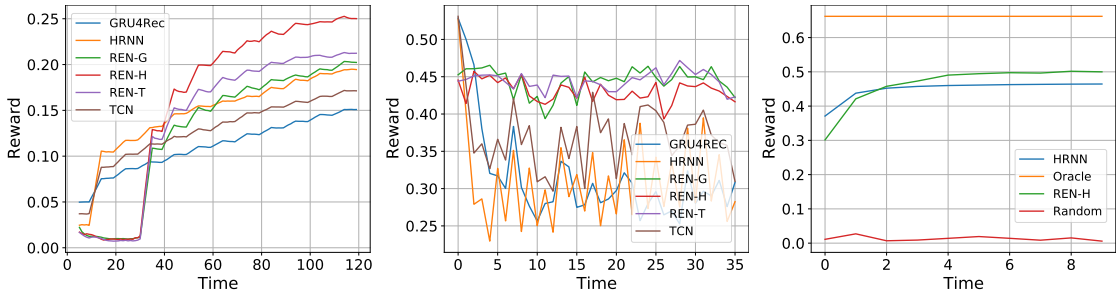

Figure 3: Rewards (precision@10, MRR, and recall@100, respectively) over time on *MovieLens-1M* (**left**), *Trivago* (**middle**), and *Netflix* (**right**). One time step represents 10 recommendations to a user, one hour of data, and 100 recommendations to a user for *MovieLens-1M*, *Trivago*, and *Netflix*, respectively.

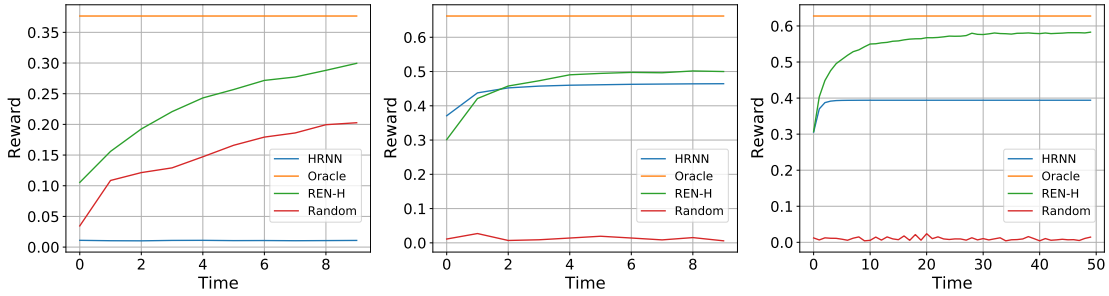

Figure 4: Rewards over time on *Netflix*. One time step represents 100 recommendations to a user.

suffers from a drastic drop in rewards because agents are allowed to recommend *only* 10 *items*, and they choose to focus only on relevance. This will inevitably ignores valuable items and harms the accuracy. In contrast, REN variants (e.g., REN-G) can effectively balance relevance and exploration for these 10 recommended items at each time step, achieving higher long-term rewards. Interestingly, we also observe that REN variants have better stability in performance compared to RNN baselines.

Netflix. Finally, we also use *Netflix*[5] to evaluate how REN performs in the slate recommendation setting and without finetuning in each time step, i.e., skipping Step (3) in Sec. 12. We pretrain REN on data from half the users and evaluate on the other half. At each time step, REN generates 100 mutually diversified items for one slate following Eqn. 5, with $p_{k,t}$ updated after every item generation. Fig. 3(right) shows the recall@100 as the reward for different methods, demonstrating REN's promising exploration ability when no finetuning is allowed (more results in the Supplement). Fig. 4(left) shows similar trends with recall@100 as the reward on the same holdout item set. This shows that the collected set contributes to building better user embedding models. Fig. 4(middle) shows that the additional exploration power comes without significant harms to the user's immediate rewards on the exploration set, where the recommendations are served. In fact, we used a relatively large exploration coefficient, $\lambda_d = \lambda_u = 0.005$, which starts to affect recommendation results on the sixth position. By additional hyperparameter tuning, we realized that to achieve better rewards on the exploration set, we may choose smaller $\lambda_d = 0.0007$ and $\lambda_u = 0.0008$. Fig. 4(right) shows significantly higher recalls close to the oracle performance, where all of the users' histories are known and used as inputs to predict the top-100 personalized recommendations.[6] Note that, for fair presentation of the tuned results, we switched the exploration set and the holdout set and used a different test user group, consisting of 1543 users. We believe that the tuned results are generalizable with new users and items, but we also realize that the *Netflix* dataset still has a significant popularity bias and therefore we recommend using larger exploration coefficients with real online systems. The inference cost is 175 milliseconds to pick top-100 items from 8000 evaluation items. It includes 100 sequential linear function solutions with 50 embedding dimensions, which is further improvable by selecting multiple items at a time in slate generation.

Conclusion

We propose the REN framework to balance relevance and exploration during recommendation. Our theoretical analysis and empirical results demonstrate the importance of considering uncertainty in the learned representations for effective exploration and improvement on long-term rewards. We provide an upper confidence bound on the estimated rewards along with its corresponding regret bound and show that REN can achieve the same rate-optimal sublinear regret even in the presence of uncertain representations. Future work could investigate the possibility of learned uncertainty in representations, extension to Thompson sampling, nonlinearity of the reward w.r.t. $\boldsymbol{\theta}_t$, and applications beyond recommender systems, e.g., robotics and conversational agents.

[5] https://www.kaggle.com/netflix-inc/netflix-prize-data

[6] The gap between oracle and 100% recall lies in the model approximation errors.

Acknowledgement

The authors thank Tim Januschowski, Alex Smola, the AWS AI's Personalize Team and ML Forecast Team, as well as the reviewers/SPC/AC for the constructive comments to improve the paper. We are also grateful for the RecoBandits package provided by Bharathan Blaji, Saurabh Gupta, and Jing Wang to facilitate the simulated environments. HW is partially supported by NSF Grant IIS-2127918 and an Amazon Faculty Research Award.

References

[1] Agarwal, A.; Hsu, D. J.; Kale, S.; Langford, J.; Li, L.; and Schapire, R. E. 2014. Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits. In *ICML*, 1638–1646.

[2] Antikacioglu, A.; and Ravi, R. 2017. Post processing recommender systems for diversity. In *KDD*, 707–716.

[3] Auer, P. 2002. Using Confidence Bounds for Exploitation-Exploration Trade-offs. *JMLR*, 3: 397–422.

[4] Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *CoRR*, abs/1803.01271.

[5] Belletti, F.; Chen, M.; and Chi, E. H. 2019. Quantifying Long Range Dependence in Language and User Behavior to improve RNNs. In *KDD*, 1317–1327.

[6] Bello, I.; Kulkarni, S.; Jain, S.; Boutilier, C.; Chi, E.; Eban, E.; Luo, X.; Mackey, A.; and Meshi, O. 2018. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019*.

[7] Chen, M.; Beutel, A.; Covington, P.; Jain, S.; Belletti, F.; and Chi, E. H. 2019. Top-k off-policy correction for a REINFORCE recommender system. In *WSDM*, 456–464.

[8] Cho, K.; van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*, 1724–1734.

[9] Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. 2011. Contextual bandits with linear payoff functions. In *AISTATS*, 208–214.

[10] Fang, H.; Zhang, D.; Shu, Y.; and Guo, G. 2019. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *arXiv preprint arXiv:1905.01997*.

[11] Foster, D. J.; Agarwal, A.; Dudík, M.; Luo, H.; and Schapire, R. E. 2018. Practical Contextual Bandits with Regression Oracles. In *ICML*, 1534–1543.

[12] Friedland, S.; and Gaubert, S. 2013. Submodular spectral functions of principal submatrices of a hermitian matrix, extensions and applications. *Linear Algebra and its Applications*, 438(10): 3872–3884.

[13] Gupta, S.; Wang, H.; Lipton, Z.; and Wang, Y. 2021. Correcting exposure bias for link recommendation. In *ICML*.

[14] Harper, F. M.; and Konstan, J. A. 2016. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4): 19.

[15] Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

[16] Hiranandani, G.; Singh, H.; Gupta, P.; Burhanuddin, I. A.; Wen, Z.; and Kveton, B. 2019. Cascading Linear Submodular Bandits: Accounting for Position Bias and Diversity in Online Learning to Rank. In *UAI*, 248.

[17] Jun, K.-S.; Willett, R.; Wright, S.; and Nowak, R. 2019. Bilinear Bandits with Low-rank Structure. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 3163–3172. PMLR.

[18] Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

[19] Korda, N.; Szorenyi, B.; and Li, S. 2016. Distributed clustering of linear bandits in peer to peer networks. In *ICML*, 1301–1309.

[20] Kveton, B.; Szepesvári, C.; Rao, A.; Wen, Z.; Abbasi-Yadkori, Y.; and Muthukrishnan, S. 2017. Stochastic Low-Rank Bandits. *CoRR*, abs/1712.04644.

[21] Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; and Ma, J. 2017. Neural attentive session-based recommendation. In *CIKM*, 1419–1428.

[22] Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 661–670.

[23] Li, S.; Karatzoglou, A.; and Gentile, C. 2016. Collaborative Filtering Bandits. In *SIGIR*, 539–548.

[24] Li, X.; and She, J. 2017. Collaborative Variational Autoencoder for Recommender Systems. In *KDD*, 305–314.

[25] Liu, Q.; Zeng, Y.; Mokhosi, R.; and Zhang, H. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *KDD*, 1831–1839.

[26] Ma, Y.; Narayanaswamy, M. B.; Lin, H.; and Ding, H. 2020. Temporal-Contextual Recommendation in Real-Time. In *KDD*.

[27] Mahadik, K.; Wu, Q.; Li, S.; and Sabne, A. 2020. Fast distributed bandits for online recommendation systems. In *SC*, 1–13.

[28] Nguyen, T. T.; Hui, P.-M.; Harper, F. M.; Terveen, L.; and Konstan, J. A. 2014. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *WWW*, 677–686.

[29] Osband, I.; Van Roy, B.; and Wen, Z. 2016. Generalization and exploration via randomized value functions. In *ICML*, 2377–2386.

[30] Quadrana, M.; Karatzoglou, A.; Hidasi, B.; and Cremonesi, P. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In *RecSys*, 130–137.

[31] Salakhutdinov, R.; Mnih, A.; and Hinton, G. E. 2007. Restricted Boltzmann machines for collaborative filtering. In *ICML*, volume 227, 791–798.

[32] Shi, J.-C.; Yu, Y.; Da, Q.; Chen, S.-Y.; and Zeng, A.-X. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In *AAAI*, volume 33, 4902–4909.

[33] Tang, J.; Belletti, F.; Jain, S.; Chen, M.; Beutel, A.; Xu, C.; and H. Chi, E. 2019. Towards neural mixture recommender for long range dependent user sequences. In *WWW*, 1782–1793.

[34] Tschiatschek, S.; Djolonga, J.; and Krause, A. 2016. Learning Probabilistic Submodular Diversity Models Via Noise Contrastive Estimation. In *AISTATS*, 770–779.

[35] van den Oord, A.; Dieleman, S.; and Schrauwen, B. 2013. Deep content-based music recommendation. In *NIPS*, 2643–2651.

[36] Vanchinathan, H. P.; Nikolic, I.; Bona, F. D.; and Krause, A. 2014. Explore-exploit in top-N recommender systems via Gaussian processes. In *RecSys*, 225–232.

[37] Wang, H. 2017. *Bayesian Deep Learning for Integrated Intelligence: Bridging the Gap between Perception and Inference*. Ph.D. thesis, Hong Kong University of Science and Technology.

[38] Wang, H.; Shi, X.; and Yeung, D. 2015. Relational stacked denoising autoencoder for tag recommendation. In *AAAI*, 3052–3058.

[39] Wang, H.; Shi, X.; and Yeung, D.-Y. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *NIPS*, 415–423.

[40] Wang, H.; Wang, N.; and Yeung, D. 2015. Collaborative deep learning for recommender systems. In *KDD*, 1235–1244.

[41] Wang, H.; and Yeung, D.-Y. 2016. Towards Bayesian deep learning: A framework and some existing methods. *TDKE*, 28(12): 3395–3408.

[42] Wang, H.; and Yeung, D.-Y. 2020. A Survey on Bayesian Deep Learning. *ACM Computing Surveys (CSUR)*, 53(5): 1–37.

[43] Wilhelm, M.; Ramanathan, A.; Bonomo, A.; Jain, S.; Chi, E. H.; and Gillenwater, J. 2018. Practical diversified recommendations on youtube with determinantal point processes. In *CIKM*, 2165–2173.

[44] Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; and Tan, T. 2019. Session-based recommendation with graph neural networks. In *AAAI*, volume 33, 346–353.

[45] Yue, Y.; and Guestrin, C. 2011. Linear Submodular Bandits and their Application to Diversified Retrieval. In *NIPS*, 2483–2491.

[46] Zhou, D.; Li, L.; and Gu, Q. 2019. Neural Contextual Bandits with Upper Confidence Bound-Based Exploration. *arXiv preprint arXiv:1911.04462*.