Generative Slate Recommendation with Reinforcement Learning

Romain Deffayet
Naver Labs Europe
Meylan, France
University of Amsterdam
Amsterdam, The Netherlands
romain.deffayet@naverlabs.com

Thibaut Thonet
Naver Labs Europe
Meylan, France
thibaut.thonet@naverlabs.com

Jean-Michel Renders
Naver Labs Europe
Meylan, France
jean-michel.renders@naverlabs.com

Maarten de Rijke
University of Amsterdam
Amsterdam, The Netherlands
m.derijke@uva.nl

ABSTRACT

Recent research has employed reinforcement learning (RL) algorithms to optimize long-term user engagement in recommender systems, thereby avoiding common pitfalls such as user boredom and filter bubbles. They capture the sequential and interactive nature of recommendations, and thus offer a principled way to deal with long-term rewards and avoid myopic behaviors. However, RL approaches are intractable in the slate recommendation scenario – where a list of items is recommended at each interaction turn – due to the combinatorial action space. In that setting, an action corresponds to a slate that may contain any combination of items.

While previous work has proposed well-chosen decompositions of actions so as to ensure tractability, these rely on restrictive and sometimes unrealistic assumptions. Instead, in this work we propose to encode slates in a continuous, low-dimensional latent space learned by a variational auto-encoder. Then, the RL agent selects continuous actions in this latent space, which are ultimately decoded into the corresponding slates. By doing so, we are able to (i) relax assumptions required by previous work, and (ii) improve the quality of the action selection by modeling full slates instead of independent items, in particular by enabling diversity. Our experiments performed on a wide array of simulated environments confirm the effectiveness of our generative modeling of slates over baselines in practical scenarios where the restrictive assumptions underlying the baselines are lifted. Our findings suggest that representation learning using generative models is a promising direction towards generalizable RL-based slate recommendation.

CCS CONCEPTS

• **Information systems** → **Recommender systems**.

KEYWORDS

Slate recommendation, Reinforcement learning, Variational auto-encoder

ACM Reference Format:
Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten de Rijke. 2023. Generative Slate Recommendation with Reinforcement Learning. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM '23), February 27-March 3, 2023, Singapore, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3539597.3570412

WSDM '23, February 27-March 3, 2023, Singapore, Singapore

ACM ISBN 978-1-4503-9407-9/23/02…$15.00
https://doi.org/10.1145/3539597.3570412

1 INTRODUCTION

Ubiquitous in online services, recommender systems (RSs) play a key role personalization by catering to users' identified tastes. Ideally, they also diversify their offerings and help users discover new interests [19]. In the latter case, RSs take on an active role, which means that recommendations influence future user behavior, and therefore their effects on users must be explicitly controlled. Such effects can be detrimental: users may get bored if too many similar recommendations are made, and it has been well-documented that users can end up in so-called filter bubbles or echo chambers [4, 13, 28]. From the perspective of the online platform or the content provider, user boredom leads to poor retention and conversion rates [17], while filter bubbles raise fairness and ethical issues for which providers can be held accountable [26]. Conversely, RSs can also positively impact users, for example, when users get interested in new, unexpected topics or when the RS offers a fair representation of available options [1]. It is natural, therefore, to balance exploitation (i.e., sticking to the known interests of the user) and exploration (i.e., further probing the user's interests) so as to avoid always recommending similar items, and encourage recommendations that boost future engagement. The reinforcement learning (RL) literature has proposed models and algorithms that aim to optimize long-term metrics by acknowledging the causal effect of recommendations on users [8, 36].

In this work we consider the common scenario of slate recommendation [8, 18, 31], which comes with specific challenges. At each interaction turn, a slate recommender system recommends a list of items from the collection, and the user interacts with zero, one or several of those items. As a consequence, users may not examine all the recommended items, which leads to biases in the observed

interactions along with a complex interplay between items in the same slate [27]. More importantly, the size of the action space, i.e., the number of possible slates, prohibits the use of off-the-shelf RL approaches [12]. Indeed, as slate recommendation is a combinatorial problem, the evaluation of all actions by the RL agent through trial and error is simply intractable: even with as few as 1,000 items in the collection, the number of possible slates of size 10 is approximately 9.6×10^{29}. We propose to tackle this problem in the context of a practical scenario, **(S)**, which fits the second-stage ranking phase [11] of many content recommendation platforms:

(**S**) *The collection contains around a thousand items, and at each turn of interaction the proposed model must select and rank 10 items to be presented to the user.*

All our tractability and feasibility statements in this paper must therefore be understood through the lens of this scenario **(S)**.

To reduce the prohibitively large size of the combinatorial action space, previous studies have proposed to decompose slates in a tractable manner [8, 18, 31] – but at the cost of restrictive assumptions, e.g., concerning mutual independence of items in the slate, knowledge of the user click model, availability of high-quality item embeddings, or that at most one item per slate is clicked.

In contrast, in this work we propose to first learn a continuous, low-dimensional latent representation of actions (i.e., slates), and then let the agent take actions within this latent space during its training phase. In practice, we obtain the latent representations by introducing a *generative modeling of slates* (GeMS) based on a variational auto-encoder (VAE) pre-trained on a dataset of observed slates and clicks, collected from a previous version of the recommender system. Such a dataset is usually available in industrial recommendation settings. Therefore, we do not rely on restrictive assumptions, and the fact that we represent full slates enables the agent to improve the quality of its recommendations, instead of using individual item representations.

Our contributions can be summarized as follows:

- We propose GeMS, a novel way to represent actions in RL for slate recommendation, by pre-training a VAE on slates and associated clicks. Unlike previous methods, GeMS is free of overly restrictive assumptions and only requires logged interaction data.
- We provide a unified terminology to classify existing slate recommendation approaches based on their underlying assumptions.
- We show on a wide array of simulated environments that previous methods underperform when their underlying assumptions are lifted (i.e., in practical settings), while GeMS allows us to recover highly rewarding policies without restrictive assumptions.
- To support the reproducibility of this work, we publicly release the code for our approach, baselines and simulator.[1]

2 RELATED WORK

Long-term user engagement. Several studies have documented the misalignment between short-term benefits and long-term user engagement [1, 17], as well as the tendency of traditional recommender systems to be detrimental to long-term outcomes [29]. Such myopic behavior is known to cause boredom and decrease user retention [1], which is prejudicial for both users and content providers. This behavior also raises concerns such as the rich-get-richer issue [8] and feeding close-mindedness [29]. Some previous studies tried to counter this effect by explicitly maximizing diversity [33] or by finding metrics correlated with long-term outcomes [2, 7]. In contrast, in our work we directly optimize long-term metrics by using reinforcement learning algorithms [8, 16, 36].

Reinforcement learning for slate recommendation. The problem of slate recommendation with reinforcement learning (RL) has been tackled in several previous studies, although the settings in which solutions were tested vary and are sometimes not applicable to our scenario **(S)**. Chen et al. [8] and Bai et al. [3] assume a simple user click model and independence of items within a slate in order to reduce the problem to choosing individual items, which they solve with the REINFORCE algorithm on a SoftMax policy. Ie et al. [18] assume knowledge of the user's click model and item relevance, which allows them to perform combinatorial optimization for the computation of Q-values. Sunehag et al. [31] take a continuous action in the product space of item embeddings, i.e., one embedding per slot in the slate, and pre-select nearest-neighbor items for full-slate Q-function evaluation. Chen et al. [9] use properties of the optimal Q-function to propose an elegant decomposition of it and generate optimal slates autoregressively. We detail the assumptions made by each of these approaches in Section 4, but we had to discard [9] due to its prohibitively heavy computation: it requires a number of neural network forward passes proportional to the slate size times the number of items in the collection (i.e., 10,000 passes in scenario **(S)**), for each training or inference step.

Our proposed approach differs from previous work because we do not manually decompose the slates using tractable heuristics based on restrictive assumptions, but instead approximate the slate generation process with a deep generative model. Our proposed framework only has a single requirement, viz. the availability of logged data with slates and associated clicks, as we will detail in Section 4. The latter assumption is by no means restrictive as such logged data is readily available in common industrial recommendation settings.

Latent action representations. While learning a latent representation of states is very common in the RL literature [14, 30], few studies have tackled the problem of latent action representation. Chandak et al. [6] train an action generation function in a supervised manner, by learning to predict the action taken from a pair of successive states. This is not directly applicable in our case, because the true user state is not observable and successive observations are simply clicks that appear to be too weak of a signal to infer the slates leading to these clicks. Botteghi et al. [5] learn a state-action world model and jointly train latent state and action representations in a model-based fashion.

Learning a world model in our setting essentially amounts to the latent modeling of slates and clicks (similar to our approach), while also conditioning on an internal hidden state.[2] The work by Zhou et al. [35] is perhaps the closest work to ours, as it uses a variational auto-encoder (VAE) to embed actions into a controllable latent space before training an RL agent. However, it does not consider slates but only simple, atomic actions. In contrast, Jiang et al. [20], Liu et al. [25] train VAEs to represent slates and their associated clicks,

[1] https://github.com/naver/gems.

[2] We tried a similar method in pilot experiments, but the additional conditioning only deteriorated the results, so we only present the condition-free method in this paper.

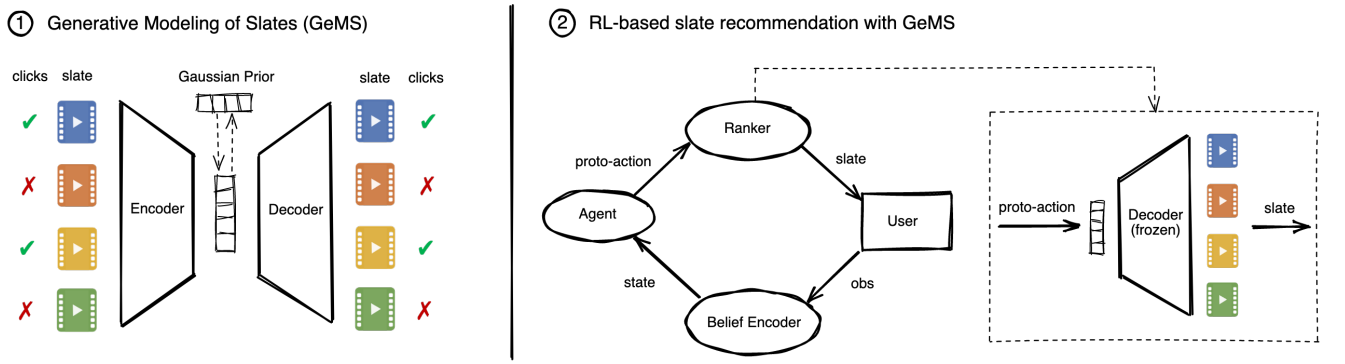


Figure 1: Our proposed framework for slate recommendation with reinforcement learning. We first pretrain our GeMS model on previously collected logged data composed of slates and associated clicks (left), then we use the frozen decoder of GeMS to decode the RL agent's low-dimensional proto-action vector into a slate (right).

but they do not investigate training an RL agent from the learned latent representation.

To the best of our knowledge, we are the first to learn a latent representation of slates for RL-based recommendation.

3 METHOD

3.1 Notations and problem definition

We consider a slate recommendation scenario in which a user interacts with a recommender system (RS) throughout an episode of T turns. At every turn $t \in \{1, \dots, T\}$, the system recommends a slate $a_t = (i_t^1, \dots, i_t^k)$ where $(i_t^j)_{1 \leqslant j \leqslant k}$ are items from the collection $\mathcal{I}$ and k is the size of the slate set by the RS designer. The user can click on zero, one or several items in the slate and the resulting click vector $c_t = (c_t^1, \dots, c_t^k), c_t^j \in \{0, 1\}$ is returned to the RS.

The problem of maximizing the cumulative number of clicks over an episode can be modeled as a partially observable Markov decision process (POMDP) $\mathcal{M}^P = (\mathcal{S}, \mathcal{O}, \mathcal{A}, R, T, \Omega)$ defined by:

- A set of states $\mathcal{S}$, which represent the unobservable state of the user's mind;
- A set of observations $\mathcal{O}$ accessible to the system. Here, observations are clicks from the previous interaction ($o_t = c_{t-1}$) and therefore lie in the space of binary vectors of size k: $\mathcal{O} = \{0, 1\}^k$;
- A set of actions $\mathcal{A}$, which is the set of all possible slates composed of items from the collection, i.e., $|\mathcal{A}| = \frac{|I|!}{(|I|-k)!}$;
- A reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, which we set to $R(s_t, a_t) = r_t = \sum_{j=1}^{k} c_t^j$ in order to reflect our long-term objective of maximizing the cumulative number of clicks; and
- A set of unknown transition and observation probabilities, respectively $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and $\Omega : \mathcal{S} \times \mathcal{A} \times \mathcal{O} \to [0, 1]$, as well as a distribution over initial states $S^1 : \mathcal{S} \to [0, 1]$.

Due to the unobserved nature of the true user state in the POMDP, it is common to train agents by relying on a proxy of the state inferred from available observations. The function that provides such proxy is traditionally referred to as the *belief encoder* [21]. We also define the concepts of a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ and trajectory $\tau = (o_t, a_t, r_t)_{1 \leqslant t \leqslant T}$. In the remainder, we write $\tau \sim \pi$ to signify that we obtain a trajectory by first sampling an initial state s_1 from S^1 and then recursively sampling actions $T - 1$ times from the policy π. The goal can now be formulated as finding an optimal policy, i.e., a policy maximizing the *expected return* $\pi^* \in \arg\max_\pi \mathbb{E}_{\tau \sim \pi} [\mathcal{R}(\tau)]$ with $\mathcal{R}(\tau) = \sum_{t=1}^{T} r_t$. Finally, given a state s and action a, we define the Q-function $Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi, s_1 = s, a_1 = a} [\mathcal{R}(\tau)]$ and V-function $V^\pi(s) = \mathbb{E}_{a \sim \pi(s)} [Q^\pi(s, a)]$.

3.2 Overview of the framework

In our proposed framework, the interactions with the environment, i.e., the user, can be described by the following repeated steps:

(1) The *belief encoder* summarizes the history of interactions with the user into a state vector;
(2) The *agent* selects a proto-action based on this state; and
(3) The *ranker* (here resulting from a VAE model) decodes this proto-action into a slate that is served to the user.

In the remainder of this section, we first detail our proposed *generative modeling of slates* (GeMS). GeMS is a deep generative model that learns a low-dimensional latent space for slates and associated clicks – thus constituting a convenient proto-action space for the RL agent and allowing for tractable RL without resorting to restrictive assumptions as in prior work [3, 8, 18, 31]. Then we describe how GeMS is integrated as a ranker in our RL framework and we briefly discuss the remaining RL components. This two-step process is depicted in Figure 1.

3.3 Generative Modeling of Slates (GeMS)

In order to instantiate our GeMS model, we propose to train a variational auto-encoder (VAE) on a precollected dataset $\mathcal{D}$ of logged interactions, as illustrated in Figure 1 (left). A VAE aims to learn a joint distribution over data samples (i.e., slates and clicks denoted as a and c, respectively) and latent encodings (i.e., proto-actions denoted as z) [22]. To do so, a parameterized distribution $p_\theta(a, c, z)$ is trained to maximize the marginal likelihood of the data $p_\theta(a, c) = \int_z p_\theta(a, c, z) dz$. In practice, due to the intractability of this integral, a parameterized distribution $q_\phi(z|a, c)$ is introduced as a variational approximation of the true posterior $p_\theta(z|a, c)$ and the VAE is trained by maximizing the evidence lower bound (ELBO):

$$\mathcal{L}_{\theta,\phi}^{\text{ELBO}} = \mathbb{E}_{a,c \sim \mathcal{D}} \left[\mathbb{E}_{z \sim q_\phi(\cdot|a,c)} [\log p_\theta(a, c|z)] - \text{KL} \left[q_\phi(z|a, c) \| p(z) \right] \right], \quad (1)$$

where $p(z)$ is the prior distribution over the latent space, KL is the Kullback-Leibler divergence [24], and z is a sample from a Gaussian distribution obtained using the reparameterization trick [22]. The distributions $q_\phi(z|a,c)$ and $p_\theta(a,c|z)$ are usually referred to as the encoder and the decoder, respectively.

The downstream performance of the RL agent we wish to ultimately learn clearly depends on the upstream ability of the VAE to properly reconstruct slates. However, as Liu et al. [25] observe, an accurate reconstruction of slates may limit the agent's capacity to satisfy the user's interests. Indeed, finding high-performance continuous control policies requires smoothness and structure in the latent space, which may be lacking if too much emphasis is given to the reconstruction objective in comparison to the prior matching objective enforced by the KL-divergence. Therefore, it is necessary to balance reconstruction and controllability, which is done by introducing an hyperparameter β as weight for the KL term in Eq. 1. Moreover, in order to promote additional structure in the latent space, we add a click reconstruction term in the loss: slates with similar short-term outcomes (i.e., clicks) are grouped together during pre-training. Yet, we may want to avoid biasing the learned representations towards click reconstruction too much, as it may come at the cost of quality of the slate reconstruction. Therefore, we introduce a hyperparameter λ to adjust this second trade-off. We show the empirical impact of β and λ in Section 6.3.

In our implementation, the prior $p(z)$ is set as a standard Gaussian distribution $\mathcal{N}(\mathbf{0},\mathbf{I})$. The encoder $q_\phi(z|a,c)$ is a Gaussian distribution with diagonal covariance $\mathcal{N}(\mu_\phi(a,c), \text{diag}(\sigma^2_\phi(a,c)))$, parameterized by a multi-layer perceptron (MLP). This MLP inputs the concatenation of learnable item embeddings and associated clicks over the whole slate, and outputs $(\mu_\phi(a,c), \log\sigma_\phi(a,c))$. For the decoder $p_\theta(a,c|z)$, another MLP takes as input the latent sample z, and outputs the concatenation of reconstructed embeddings $\mathbf{e}^j_\theta(z)$ and click probabilities $p^{j,c}_\theta(c_j|z)$ for each slot j in the slate. We then derive logits for the item probabilities $p^{j,a}_\theta(a_j|z)$ by taking the dot-product of the reconstructed embedding $\mathbf{e}^j_\theta(z)$ with the embeddings of all items in the collection. For collection items, we use the current version of embeddings learned within the encoder, but we prevent the gradient from back-propagating to them using the stop-gradient operator to avoid potential degenerate solutions.

In summary, the VAE is pre-trained by maximizing the ELBO on the task of reconstructing slates and corresponding clicks, i.e., by minimizing $\mathcal{L}^{\text{GeMS}}_{\theta,\phi} = \mathbb{E}_{a,c\sim\mathcal{D}}[\mathcal{L}^{\text{GeMS}}_{\theta,\phi}(a,c)]$ with:

$$\mathcal{L}^{\text{GeMS}}_{\theta,\phi}(a,c) = \overbrace{\sum_{j=1}^{k} \log p^{j,a}_\theta(a_j|z_\phi(a,c))}^{\text{slate reconstruction}} + \overbrace{\lambda \sum_{j=1}^{k} \log p^{j,c}_\theta(c_j|z_\phi(a,c))}^{\text{click reconstruction}} + \overbrace{\beta \sum_{i=1}^{d} \left(\sigma^2_{\phi,i} + \mu^2_{\phi,i} - \log\sigma_{\phi,i} - 1\right)}^{\text{KL-divergence}} \tag{2}$$

where $z_\phi(a,c) = \mu_\phi(a,c) + \text{diag}(\sigma^2_\phi(a,c)) \cdot \epsilon$, for $\epsilon \sim \mathcal{N}(\mathbf{0},\mathbf{I})$. Here, d is the dimension of the latent space, and β and λ are hyperparameters controlling the respective weight of the KL term and the click reconstruction term. Note that the KL term takes this simple form due to the Gaussian assumption on $q_\phi(z|a,c)$ and the $\mathcal{N}(\mathbf{0},\mathbf{I})$ prior.

3.4 RL agent and belief encoder

After the pre-training step described in Section 3.3, the parameters of GeMS are frozen and we use its decoder as the ranker in our RL framework. The RL agent can then be trained to maximize the discounted return by taking proto-actions within the VAE's latent space. To generate a slate $(i^1,\dots,i^k)$ from the agent's proto-action z, we take for each slot $j \in \{1,\dots,k\}$ the most likely item according to the decoder: $i^j = \arg\max_{i\in\mathcal{I}} p^{j,a}_\phi(i|z)$.

Since our focus within the RL framework is on the choice of the ranker, we adopt a standard implementation of the belief encoder and the agent: the former is modeled by a gated recurrent unit (GRU) [10] taking as input the concatenation of item embeddings and respective clicks from each slate, and the latter is a soft actor-critic (SAC) [15] algorithm. We chose SAC because it is a well-established RL algorithm, known for its strong performance and data-efficiency in continuous control. Additionally, SAC adds an entropy term incentivizing exploration which we have noticed during our experiments to be important to attain high performance in highly stochastic recommendation environments.

4 BASELINES AND THEIR ASSUMPTIONS

We evaluate our proposed method against four main baselines derived from prior work. In this section, we describe these baselines as well the assumptions on user behavior that they formulate in order to make the combinatorial problem of slate recommendation tractable. By doing so, we are able to compare the assumptions made by these baselines and highlight the generality of our method in Table 1. Note that we only report from previous studies the mechanism used for slate generation, which is the topic of this study, and ignore other design choices.

SoftMax. In [3, 8], the authors reduce the combinatorial problem of slate optimization to the simpler problem of item optimization: the policy network output is a softmax layer over all items in the collection, and items are sampled with replacement to form slates. Doing so requires the mild assumption that *the Q-value of the slate can be linearly decomposed into item-specific Q-values* (**DQ**). But more importantly, it also requires two strong assumptions, namely *users can click on at most one item per slate* (**1CL**) and *the returns of items in the same slate are mutually independent* (**MI**). Together, these assumptions are restrictive, because their conjunction means that the click probability of an item in the slate does not depend on the item itself. Indeed, having dependent click probabilities (to enforce the single click) and independent items in the slate is compatible only if click probabilities do not depend on items.

SlateQ. Ie et al. [18] propose a model-based approach in which the click behavior of the user is given, and Q-learning [34] is used to plan and approximate users' dynamic preferences. On top of the earlier DQ and 1CL, it requires *access to the true relevance and click model* (**CM**), which is an unfair advantage compared to other methods. For computational efficiency reasons, we adopt the faster variant referred to as QL-TT-TS in the original paper.

TopK. Even though, to the best of our knowledge, no work has proposed this approach, we include it in our set of baselines as it is a natural way to deal with slate recommendation. The agent takes continuous actions in the space of item embeddings, and we generate slates by taking the k items from the collection with the closest embeddings to the action, according to a similarity metric (the dot-product in practice). This method therefore assumes the *availability of logged data of past interactions* (**LD**), in order to pre-train item embeddings. In our experiments, we evaluate two variants of this baseline: *TopK (MF)*, where item embeddings are learned by matrix factorization [23], and *TopK (ideal)*, which uses ideal item embeddings, i.e., the embeddings used internally by the simulator (see Section 5.1). The latter version clearly has an unfair advantage. Also, because ranking items this way assumes that the most rewarding items should appear on top, it makes the *sequential presentation* (**SP**) assumption from [31] that *the true click model is top-down and fading*, i.e., if $c(i)$ indicates that item i has been clicked and $l \leqslant k$ is the position of i in slate a, then $P(c(i)|s,a) = P(c(i)|s,a_{\leqslant l}) \leqslant P(c(i)|s,\tilde{a}_{\leqslant l-1})$, where $a_{\leqslant l} = (i^1,\dots,i^{l-1},i)$ and $\tilde{a}_{\leqslant l-1} = (i^1,\dots,i^{l-2},i)$.

WkNN. In [31], the authors propose a finer-grained and potentially more capable variant of TopK referred to as *Wolpertinger* [12]: the agent takes actions in the product-space of item embeddings over slate slots, i.e., continuous actions of dimension $k \times d$, where d is the dimension of item embeddings. Then, for each slot in the slate, p candidate items are selected by Euclidean distance with embeddings of items from the collection, and every candidate item's contribution to the Q-value is evaluated in a greedy fashion. Besides LD and DQ, WkNN requires two strong assumptions to ensure submodularity of the Q-function: sequential presentation SP and *execution is best* (**EIB**), i.e., *recommendations that are risky on the short term are never worth it*. Formally, this translates as: $\mathbb{P}(R(s,\pi_1(s)) = 0) \geqslant \mathbb{P}(R(s,\pi_2(s)) = 0) \implies V^{\pi_1}(s) \leqslant V^{\pi_2}(s)$ for any policies π_1, π_2. Note that it partly defeats the purpose of long-term optimization.

In Table 1, we summarize the assumptions made by each baseline. In comparison to prior work, our proposed framework has a single assumption: the availability of logged data with slates and associated clicks (LD), as Table 1 indicates. This assumption is by no means restrictive as such logged data is readily available in common industrial recommendation settings.

On top of these baselines, we also include a **random** policy and a **short-term oracle** as reference points. The short-term oracle has access to the true user and item embeddings, enabling it to select the items with the highest relevance probability in each slate. Therefore, at each turn of interaction, it gives an upper bound on the immediate reward but it is unable to cope with boredom and influence phenomena.

5 EXPERIMENTAL SETUP

5.1 Simulator

We design a simulator that allows us to observe the effect of lifting the assumptions required by the baselines, and we experiment with several simulator variants to ensure generalizability. We summarize our main design choices below and refer the reader to our code

Table 1: Comparison of assumptions made by prior work. Our method only requires access to logged interaction data.

	1CL	DQ	MI	CM	SP	EIB	LD
SoftMax [3, 8]	✓	✓	✓	✗	✗	✗	✗
SlateQ [18]	✓	✓	✗	✓	✗	✗	✗
WkNN [31]	✗	✓	✗	✗	✓	✓	✓
TopK	✗	✗	✗	✗	✓	✗	✓
GeMS (Ours)	✗	✗	✗	✗	✗	✗	✓

available online[3] for a more detailed description.

Item and user embeddings. Following scenario **(S)**, our simulator includes 1,000 items. We consider a cold-start situation where users are generated on-the-fly for each new trajectory. Items and users are randomly assigned embeddings of size 20, corresponding to ten 2-dimensional topics: $\mathbf{e} = (\mathbf{e}^1,\dots,\mathbf{e}^{10})$. Each 2-dimensional vector $\mathbf{e}^t$ is meant to capture the existence of subtopics within topic t. The embedding of a user or item x is generated using the following process: (i) sample topic propensities $w_x^t \sim \mathcal{U}(0,1)$ and normalize such that $\sum_t w_x^t = 1$; (ii) sample topic-specific components $\epsilon_x^t \sim \mathcal{N}(\mathbf{0}, 0.4 \cdot \mathbf{I}_2)$ and rescale as $\mathbf{e}_x^t = w_x^t \cdot \min(|\epsilon_x^t|, 1))$; and (iii) normalize the embedding $\mathbf{e}_x = (\mathbf{e}_x^1,\dots,\mathbf{e}_x^{10})$ such that $\|\mathbf{e}_x\| = 1$. Each item is associated to a main topic, defined as $t(i) = \arg\max_{1\leqslant t\leqslant 10} \|\mathbf{e}_i^t\|$.

To accomodate different types of content and platforms, we derive two variants of item embeddings in the simulator: one with embeddings obtained as described above, and one with embeddings for which we square and re-normalize each component. In Section 6, we highlight this difference in peakedness by referring to the former as *diffuse embeddings* and the latter as *focused embeddings*.

Relevance computation. The relevance probability of item i for user u is a monotonically increasing function of the dot-product between their respective embeddings: $\text{rel}(i,u) = \sigma(\mathbf{e}_i{}^T \mathbf{e}_u)$, where σ is a sigmoid function.

Boredom and influence effects. User embeddings can be affected by two mechanisms: *boredom* and *influence*. Each item i clicked by user u influences the user embedding in the next interaction turn as: $\mathbf{e}_u \leftarrow \omega \mathbf{e}_u + (1-\omega)\mathbf{e}_i$, where we set $\omega = 0.9$ in practice. Additionally, if in the last 10 items clicked by user u five have the same main topic t^b, then u gets bored with this topic, meaning we put $\mathbf{e}_u^{t^b} = \mathbf{0}$ for 5 turns. These mechanisms have been defined to penalize myopic behavior and encourage long-term strategies.

Click model. Users click on recommended items according to a position-based model, i.e., the click probability is the product of item-specific attractiveness and rank-specific examination probabilities: $\mathbb{P}(c|i,r) = A_i \times E_r$. Specifically, we define for an item located at rank r: $E_r = v\varepsilon^r + (1-v)\varepsilon^{k+1-r}$ with $\varepsilon = 0.85$. It is a mixture of the terms ε^r and ε^{k+1-r}, which respectively capture the top-down and bottom-up browsing behaviors. We use two variants of this click model in our experiments: *TopDown* with $v = 1.0$ and *Mixed* with $v = 0.5$. The attractiveness of an item is set to its relevance in TopDown and Mixed. In addition, we consider a third variant *DivPen* which also penalizes slates that lack diversity: A_i is down-weighted by a factor of 3 if more than 4 items from the slate have the same main topic (as in Mixed, we also set $v = 0.5$ for DivPen).

In summary, our experiments are performed on 6 simulator variants

[3] https://naver/github/gems

Table 2: Average cumulative number of clicks on the test set for our 6 simulated environments. Bold: best method; underlined: 2nd-best method; †: statistically significantly better than all other methods. 95% confidence intervals are given in parentheses. Methods grouped under "Disclosed env." have access to privileged information about the environment and can therefore not be fairly compared with "Undisclosed env." methods.

	Method	Focused item embeddings			Diffuse item embeddings		
		TopDown	Mixed	DivPen	TopDown	Mixed	DivPen
Disclosed env.	Short-term oracle	107.7	101.6	85.4	96.7	94.6	78.8
	SAC+TopK (ideal)	429.0 (±5.9)	384.1 (±13.5)	386.3 (±15.5)	373.9 (±25.0)	371.9 (±36.4)	341.3 (±55.3)
	SlateQ	206.5 (±4.1)	202.7 (±3.4)	119.0 (±3.9)	209.5 (±5.4)	192.7 (±5.1)	117.8 (±5.8)
Undisclosed env.	Random	33.8 (±0.2)	33.9 (±0.2)	33.6 (±0.2)	33.3 (±0.2)	33.2 (±0.2)	32.9 (±0.2)
	REINFORCE+SoftMax	248.1 (±19.3)	233.5 (±18.5)	249.1 (±11.6)	249.5 (±15.3)	214.7 (±25.0)	213.8 (±27.1)
	SAC+WkNN	98.5 (±8.9)	97.7 (±10.8)	95.5 (±9.9)	107.2 (±8.9)	89.8 (±7.4)	92.5 (±5.0)
	SAC+TopK (MF)	254.4 (±17.1)	232.7 (±19.4)	242.2 (±15.4)	249.7 (±10.3)	184.1 (±1.3)	231.4 (±13.3)
	SAC+GeMS (Ours)	**305.3**† (±21.9)	**242.6** (±21.5)	**254.1** (±27.7)	**300.0**† (±42.8)	**260.6**† (±27.2)	**249.6** (±37.6)

defined by the choice of item embedding peakedness (*diffuse item embeddings* or *focused item embeddings*) and the choice of click model (*TopDown*, *Mixed*, or *DivPen*).

5.2 Implementation and evaluation details

Our implementation aims to be as standard as possible, considering the literature on RL, in order to ensure reproducibility. All baselines are paired with SAC [15], except SlateQ which is based on Q-Learning [34], and SoftMax, which we pair with REINFORCE [32] because it requires a discrete action space and a discretized variant of SAC led to lower performance in our experiments. We implement all agents using two-layer neural networks as function approximators, and use target networks for Q-functions in Slate-Q and SAC. For hyperparameters common to baselines and our method, we first performed a grid search over likely regions of the space on baselines, and re-used the selected values for our method. For all methods we use the Adam optimizer with learning rates of 0.001 for Q-networks and 0.003 for policy networks when applicable, as well as a discount factor $\gamma = 0.8$ and a polyak averaging parameter $\tau = 0.002$. For the hyperparameters specific to our method (d, β and λ), we perform a grid search on the TopDown environment with focused item embeddings and select the combination with the highest validation return. This combination is then re-used on all other environments. The searched ranges were defined as $d \in \{16, 32\}$, $\beta \in \{0.1, 0.2, 0.5, 1.0, 2.0\}$ and $\lambda \in \{0.0, 0.2, 0.5, 1.0\}$.

For methods making the (LD) assumption, we generated a dataset of 100K user trajectories (with 100 interactions turns each) from an ϵ-greedy oracle policy with $\epsilon = 0.5$, i.e., each recommended item is selected either uniformly randomly or by an oracle, with equal probabilities. The VAE in GeMS is trained on this dataset for 10 epochs with a batch size of 256 and a learning rate of 0.001. For approaches requiring pre-trained item embeddings (TopK and WkNN), we learn a simple matrix factorization model on the generated dataset by considering as positive samples the pairs composed of the user in the trajectory and each clicked item in their recommended slates.

In all of our experiments, we compare average cumulative rewards over 10 seeded runs, corresponding to ten initializations of the agent's parameters. In the case of GeMS, the seed also controls the initialization of the VAE model during pre-training. We train agents for 100K steps. Each step corresponds to a user trajectory, composed of 100 interaction turns (i.e., 100 slates successively presented to the user) for a unique user. Every 1,000 training steps, we also evaluate the agents on 200 validation user trajectories. Finally, the agents are tested by selecting the checkpoint with the highest validation return and applying it on 500 test user trajectories. Confidence intervals use Student's t-distribution, and statistical tests are Welch's t-test. Both are based on a 95% confidence level.

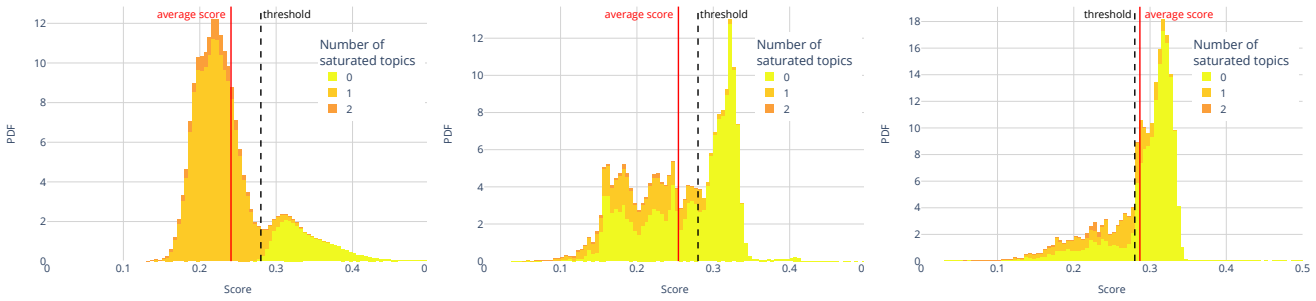
6 RESULTS

In our experiments, we investigate the following research questions: (**RQ1**) How does our slate recommendation framework based on GeMS compare to previous methods when the underlying assumptions of the latter are lifted? (**RQ2**) Does the proposed GeMS framework effectively balance immediate and future rewards to avoid boredom? (**RQ3**) How do the balancing hyperparameters β and λ in GeMS impact the downstream RL performance?

6.1 Comparison of our method against baselines (RQ1)

In this section, we compare the performance of our method and baselines on a wide array of simulated environments, corresponding to the six environments described in Section 5.1.

Overview of the results. Table 2 shows the average test return (i.e., cumulated reward or cumulated number of clicks) after training on 100K user trajectories. We group methods into two categories: *Disclosed env.*, i.e., methods leveraging hidden environment information, and *Undisclosed env.*, i.e., methods that consider the environment as a black-box and are therefore practically applicable. A first observation we can draw, regardless of the specific environment used, is that the short-term oracle is easily beaten by most approaches. Indeed, the simulator penalizes short-sighted recommendations that lead to boredom: in these environments, *diversity is required to reach higher returns.* We can also observe the superiority of SAC+TopK (Ideal). This is not surprising, as this method benefits from an unfair advantage – access to true item embeddings – but it suggests that practically applicable methods could be augmented with domain knowledge to improve their performance. However, despite having access to privileged information, SlateQ's performance is subpar, especially in DivPen environments. Its lower performance might be explained by its approximate optimization



(a) Short-term oracle. (b) SAC+GeMS with $\gamma = 0$. (c) SAC+GeMS with $\gamma = 0.8$.

Figure 2: Distribution of the relevance scores of items recommended by (a) a short-term oracle, (b) SAC+GeMS with $\gamma = 0$ and (c) SAC+GeMS with $\gamma = 0.8$. Boredom penalizes item scores and is visualized by orange areas. The myopic approaches (left, center) lead to more boredom than the long-term approach (right), and therefore to lower average item scores (solid red lines).

strategy and restrictive single-click assumption.

Overall comparison of methods. *The proposed SAC+GeMS compares favorably to baselines across the range of environments we simulate.* Out of the 6 tested environments, SAC+GeMS obtained the best average results on all of them, among which 3 show a statistically significant improvement over all other methods. SAC+WkNN performs very poorly: we hypothesize that the approach suffers from the curse of dimensionality due to the larger action space (200 dimensions in our experiments) and the assumption made by the approach that candidate items need to be close to target item embeddings according to the Euclidean distance. SAC+TopK (MF) is more competitive, but the large difference with SAC+TopK (ideal) suggests that TopK is very sensitive to the quality of item embeddings. Despite its very restrictive assumptions and lack of theoretical guarantees in our setup, REINFORCE+SoftMax was a very competitive baseline overall. However, while its best checkpoint had high return, its training was unstable and failed to converge in our experiments, which suggests it may be unreliable.

Comparisons across environments. The TopDown environment is the easiest for most methods, regardless of the type of item embeddings. This is not surprising as all methods besides Random either assume a top-down click model, sample items in a top-down fashion or rely on data from a top-down logging policy. However, it is worth noting that other factors can dominate the performance, such as sub-optimality of item embeddings for SAC+TopK (MF). Conversely, DivPen was harder for most methods, because it requires a strong additional constraint to obtain high returns: intra-slate diversity must be high. SAC+GeMS was also affected by these dynamics, but remained able to beat other methods by generating diverse slates. Finally, the use of diffused item embeddings does not appear to cause lower returns for GeMS, compared with focused ones, but is associated with larger confidence intervals for SAC+GeMS: indeed, pivot items spanning multiple topics are more likely to be attractive, at the expense of more fine-grained strategies, making the training process uncertain.

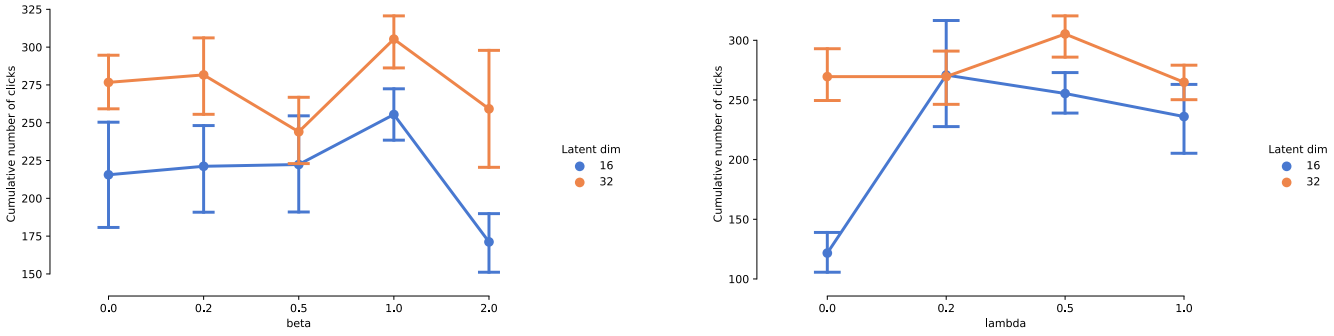
6.2 GeMS overcomes boredom to improve its return (RQ2)

In Section 1 we highlighted that long-term optimization with RL can penalize myopic behavior such as recommending only highly relevant but similar items, which may lead to boredom. In this section, we verify that SAC+GeMS is able to adapt its slate selection to cope with boredom. We recall that in our simulated environments (detailed in Section 5.1), users get bored of a particular topic whenever 5 of their latest 10 clicks were on items from that topic. When a topic is saturated, its corresponding dimensions in the user embedding are set to **0**, which has the effect of diminishing the attractiveness of future items presented to the user. It is therefore necessary to avoid boredom in order to reach higher returns, even if it comes at the cost of lower immediate rewards.

In this section, we compare three approaches on the TopDown environment with focused item embeddings: (i) the short-term oracle (STO) always maximizing the immediate reward, (ii) SAC+GeMS with $\gamma = 0.8$ (i.e., our proposed method) where γ is the discount factor of the RL algorithm, and (iii) SAC+GeMS with $\gamma = 0$ which does not explicitly include future rewards in its policy gradient. In this environment, SAC+GeMS$^{\gamma=0.8}$ achieves an average test return of 305.3, while SAC+GeMS$^{\gamma=0}$ reaches 194.3, and STO only obtains 107.7. These results suggest that long-term optimization is indeed required to reach higher returns. It may seem surprising that SAC+GeMS$^{\gamma=0}$ gets better returns than STO, but its training objective incentivizes *average* immediate rewards, which implicitly encourages it to avoid low future rewards. However, adopting an explicit mechanism to account for its causal effect on the user (i.e., setting $\gamma = 0.8$) allows SAC+GeMS to improve its decision-making.

In Figure 2, we plot the distribution of item scores (i.e., the dot-product between internal user and item embeddings as defined in Section 5.1) for the items recommended in slates by each of the three methods, with the same seed for all three plots. The dashed vertical line shows the score threshold of 0.28 needed to reach a relevance probability of 0.5. Therefore, items on the left of this line have a lower click probability while items on the right have a higher click probability. The color indicates how many topics were saturated when the agent recommended that particular item whose score is plotted: one can see that when the user is bored of at least one topic, items become less attractive as scores are reduced.

When no topic is saturated (i.e., yellow distribution), STO recommends items with excellent scores (above the threshold and up to 0.45): as a consequence, STO gets high immediate rewards. However, by doing so it incurs a lot of boredom (large orange



(a) Impact of β for $\lambda = 0.5$. **(b) Impact of λ for $\beta = 1.0$.**

Figure 3: Average cumulative number of clicks on the validation set obtained by SAC+GeMS with its best validation checkpoint, for different values of β and λ (defined in Section 3.3). We also display 95% confidence intervals.

areas). Overall, it leads to lower expected scores (solid red line) and therefore fewer clicks. Conversely, SAC+GeMS$^{\gamma=0.8}$ sacrifices some immediate reward (yellow distribution shifted to the left) but causes very little boredom (small orange area). Overall, *by trading off relevance and diversity, SAC+GeMS$^{\gamma=0.8}$ yields good immediate rewards while limiting boredom.* It therefore gets higher average scores. SAC+GeMS$^{\gamma=0}$ exhibits an intermediate behavior due to its limited capabilities: it recommends items of varying relevance, yet leads to substantial boredom (larger orange area than for $\gamma = 0.8$).

6.3 Balancing hyperparameters β and λ (RQ3)

In Section 3.3, we suggested that the choice of β and λ leads to trade-offs that may impact the downstream performance of SAC+GeMS. As a reminder, β adjusts the importance of accurate reconstruction versus smoothness and structure in the latent space (i.e., controllability), while λ weights the click reconstruction with respect to the slate reconstruction. Next, we verify our intuition on the importance of these trade-offs by reporting (in Figure 3) the best validation return obtained for different values of said hyperparameters, on the TopDown environment with focused item embeddings.

Figure 3a suggests that, indeed, there exists a "sweet spot" in the selection of β. It confirms the intuition described in Section 3.3 and the observation of Liu et al. [25]: *β must be appropriately balanced in order to ensure high performance on the downstream RL task.* Specifically, we found that choosing $\beta = 1.0$ leads to the highest return overall, regardless of whether a latent dimension of 16 or 32 is used.

The impact on the downstream performance of the trade-off between slate and click reconstruction (Figure 3b) is less prominent but can still be observed. It justifies our choice to add the click reconstruction term in the loss (Eq. 2), even though clicks output by GeMS' decoder are not used during RL training. This also confirms the importance of introducing and adjusting the hyperparameter λ: *modeling clicks jointly with slates improves the final performance of SAC+GeMS, but properly weighting the click reconstruction objective with respect to the slate reconstruction objective is necessary.*

7 CONCLUSION

We have presented GeMS, a slate representation learning method based on variational auto-encoders for slate recommendation with reinforcement learning. This method has the notable advantage of being flexible, allowing full-slate modeling and lightweight assumptions, in contrast with existing approaches.

Findings and broader impact. Our experiments across a wide array of environments demonstrate that GeMS compares favorably against existing slate representation methods in practical settings. Moreover, our empirical analysis highlights that it effectively balances immediate and future rewards, and that the trade-offs imposed by β and λ significantly impact the RL downstream performance, indicating that properly balancing these hyperparameters is critical. Our work suggests that generative models are a promising direction for representing rich actions such as slates.

Limitations. Our simulated experiments demonstrate the effectiveness of GeMS for representing slates in an RL framework. However, it is well-known that online training of RL agents is too expensive and risky, and that in practice agents must be trained offline, i.e., directly from logged data [8]. We did not address here the specific challenges of offline RL, as we wished to isolate the contribution of the slate representation to downstream performance.

Future work. In future work, we will investigate how generative models can be leveraged in the offline setting, in different scenarios, or with even richer actions. We also plan to look into improvements of the architectures used for structured action representations, for example by using domain knowledge and user models.

ACKNOWLEDGMENTS

This research was (partially) funded by the Hybrid Intelligence Center, a 10-year program funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research, https://hybrid-intelligence-centre.nl. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

REFERENCES

[1] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra, and Mounia Lalmas. 2020. Algorithmic Effects on the Diversity of Consumption on Spotify. In *WWW '20*. 2155–2165.

[2] Susan Athey, Raj Chetty, Guido W Imbens, and Hyunseung Kang. 2019. *The Surrogate Index: Combining Short-Term Proxies to Estimate Long-Term Treatment Effects More Rapidly and Precisely.* Technical Report. National Bureau of Economic Research.

[3] Xueying Bai, Jian Guan, and Hongning Wang. 2019. A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation. In *NeurIPS '19*. 10734–10745.

[4] Eytan Bakshy, Solomon Messing, and Lada Adamic. 2015. Exposure to Ideologically Diverse News and Opinion on Facebook. *Science* 348, 6239 (2015), 1130–1132.

[5] Nicolò Botteghi, Mannes Poel, Beril Sirmaçek, and Christoph Brune. 2021. Low-Dimensional State and Action Representation Learning with MDP Homomorphism Metrics. *arXiv:2107.01677* (2021).

[6] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. 2019. Learning Action Representations for Reinforcement Learning. In *ICML '19*. 941–950.

[7] Praveen Chandar, Brian St. Thomas, Lucas Maystre, Vijay Pappu, Roberto Sanchis-Ojeda, Tiffany Wu, Ben Carterette, Mounia Lalmas, and Tony Jebara. 2022. Using Survival Models to Estimate User Engagement in Online Experiments. In *WWW '22*. 3186–3195.

[8] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. 2019. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *WSDM '19*. 456–464.

[9] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative Adversarial User Model for Reinforcement Learning Based Recommendation System. In *ICML '19*. 1052–1061.

[10] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP '14*. 1724–1734.

[11] Van Dang, Michael Bendersky, and W. Bruce Croft. 2013. Two-Stage Learning to Rank for Information Retrieval. In *ECIR '13*. 423–434.

[12] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep Reinforcement Learning in Large Discrete Action Spaces. *arXiv:1512.07679* (2015).

[13] Seth R. Flaxman, Sharad Goel, and Justin M. Rao. 2016. Filter Bubbles, Echo Chambers, and Online News Consumption. *Public Opinion Quarterly* 80, S1 (2016), 298–320.

[14] David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *NeurIPS '18*. 2455–2467.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *ICML '18*. 1856–1865.

[16] Christian Hansen, Rishabh Mehrotra, Casper Hansen, Brian Brost, Lucas Maystre, and Mounia Lalmas. 2021. Shifting Consumption towards Diverse Content on Music Streaming Platforms. In *WSDM '21*. 238–246.

[17] Henning Hohnhold, Deirdre O'Brien, and Diane Tang. 2015. Focusing on the Long-Term: It's Good for Users and Business. In *KDD '15*. 1849–1858.

[18] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. 2019. SlateQ: A Tractable Decomposition for Reinforcement Learning with Recommendation Sets. In *IJCAI '19*. 2592–2599.

[19] Dietmar Jannach, Pearl Pu, Francesco Ricci, and Markus Zanker. 2021. Recommender Systems: Past, Present, Future. *AI Mag.* 42, 3 (2021), 3–6.

[20] Ray Jiang, Sven Gowal, Yuqiu Qian, Timothy A. Mann, and Danilo J. Rezende. 2019. Beyond Greedy Ranking: Slate Optimization via List-CVAE. In *ICLR '19*.

[21] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. 1998. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence* 101, 1 (1998), 99–134.

[22] Diederik Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR '14*.

[23] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37.

[24] Solomon Kullback and Richard A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.

[25] Shuchang Liu, Fei Sun, Yingqiang Ge, Changhua Pei, and Yongfeng Zhang. 2021. Variation Control and Evaluation for Generative Slate Recommendations. In *WWW '21*. 436–448.

[26] Farzan Masrour, Tyler Wilson, Heng Yan, Pang-Ning Tan, and Abdol-Hossein Esfahanian. 2020. Bursting the Filter Bubble: Fairness-Aware Network Link Prediction. In *AAAI '20*. 841–848.

[27] James McInerney, Brian Brost, Praveen Chandar, Rishabh Mehrotra, and Benjamin Carterette. 2020. Counterfactual Evaluation of Slate Recommendations with Sequential Reward Interactions. In *KDD '20*. 1779–1788.

[28] Eli Pariser. 2011. *The Filter Bubble: What the Internet Is Hiding from You*. The Penguin Press.

[29] Wilbert Samuel Rossi, Jan Willem Polderman, and Paolo Frasca. 2021. The Closed Loop between Opinion Formation and Personalised Recommendations. *IEEE Transactions on Control of Network Systems* (2021).

[30] Adam Stooke, Kimin Lee, Pieter Abbeel, and Michael Laskin. 2021. Decoupling Representation Learning from Reinforcement Learning. In *ICML '21*. 9870–9879.

[31] Peter Sunehag, Richard Evans, Gabriel Dulac-Arnold, Yori Zwols, Daniel Visentin, and Ben Coppin. 2015. Deep Reinforcement Learning with Attention for Slate Markov Decision Processes with High-Dimensional States and Actions. *arXiv:1512.01124* (2015).

[32] Richard Sutton and Andrew Barto. 2018. *Reinforcement Learning: An Introduction*. MIT Press, 326–329.

[33] Isaac Waller and Ashton Anderson. 2019. Generalists and Specialists: Using Community Embeddings to Quantify Activity Diversity in Online Platforms. In *WWW '19*. 1954–1964.

[34] Christopher Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8 (1992), 279–292.

[35] Wenxuan Zhou, Sujay Bajracharya, and David Held. 2020. PLAS: Latent Action Space for Offline Reinforcement Learning. In *CoRL '20*. 1719–1735.

[36] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement Learning to Optimize Long-Term User Engagement in Recommender Systems. In *KDD '19*. 2810–2818.