Simulating Real-World News Consumption: Deep Q-Learning for Diverse User-Centric Slate Recommendations

Aayush Singha Roy[1,2], Elias Tragos[1,2], Aonghus Lawlor[1,2] and Neil Hurley[1,2]

[1]University College Dublin, Dublin, Ireland
[2]Insight Centre of Data Analytics, Dublin, Ireland

Abstract

Tailoring recommendations to individual preferences remains a central hurdle for session-based recommendation systems. Reinforcement Learning (RL) presents a promising avenue for optimizing long-term user engagement, particularly through slate recommendation strategies. In our research, we endeavor to construct a simulation environment fuelled by real-world data for personalized news recommendations, with the overarching goal of satisfying the diversity preferences of both specialist and generalist users. To tackle this challenge, we design a RL framework to curate slates that emphasize the importance of considering user diversity in slate recommendations, thus enhancing the overall user experience. By leveraging RL algorithms, we can better assess the long-term impact of our recommendation strategies. Through our study, we aim to contribute to the advancement of RL based personalized news slate recommendation systems by evaluating our simulation based on real data, to bridge the gap between simulation and reality, ultimately enhancing user engagement and satisfaction in accessing content tailored to their individual interests and preferences.

Keywords

News recommendation, Slate recommendation, Reinforcement learning, User diversity

1. Introduction and Related Works

In recent years, recommender systems have increasingly focused on modeling user intent during sessions [1, 2, 3, 4]. A crucial aspect of understanding user behavior is their diversity quotient, reflecting the range of content they engage with. Users may lean towards closely related content, interacting with a limited portion of available content, or they may prefer diverse content, exploring various segments of the content space.

Interpreting interaction signals becomes especially hard in settings in which recommender systems are required to recommend a set of items that together serve a user's needs, commonly known as *slate recommendation* [5]. Reinforcement learning (RL) has emerged as a promising approach for slate recommendation due to its ability to estimate long-term value (LTV) [6, 7, 8]. However, considering that an action corresponds to recommending a slate of items, the combinatorially large action-space poses a severe issue to the RL algorithm. Some work, such as [9], has proposed slate recommendation based on RL, but the main limitation is scalability to real-world recommender systems with massive item catalogs. In [10], SlateQ is presented, which decomposes the state-action value of a slate into its item-wise Q-values, addressing combinatorial complexity in recommendation scenarios with large item sets [11]. Extending on this work, in [12] an algorithm is introduced to streamline Q-function evaluation, reducing inference time for real-world deployment. However, this work does not support different user profiles because learning a single item representation limits the candidate items to a single topic. In previous research on reinforcement learning-based slate recommendation, the majority

12th International Workshop on News Recommendation and Analytics in Conjunction with ACM RecSys 2024, October 18th, Bari, Italy

aayush.singharoy@insight-centre.org (A. S. Roy); elias.tragos@insight-centre.org (E. Tragos); aonghus.lawlor@insight-centre.org (A. Lawlor); neil.hurley@insight-centre.org (N. Hurley)
https://asr419.github.io/ (A. S. Roy)
0009-0000-7085-3306 (A. S. Roy); 0000-0001-9566-531X (E. Tragos); 0000-0002-6160-4639 (A. Lawlor); 0000-0001-8428-2866 (N. Hurley)

of studies have been evaluated within simulation environments, primarily due to the infeasibility of conducting live experiments. However, transferring policies learned in simulation into the real-world still remains an open research challenge.

In [13], it has been highlighted that the ecological validity of simulations is clearly limited. Moreover, evaluating solely on short-term rewards or through a myopic lens ignores the causal effects of recommendations on users, which is particularly relevant in the case of slate recommendations during a session. Firstly, the objective of this paper is to address the sim-to-real gap by developing a simulator for slate recommendation extending the RecSim[14] environment, grounded in real-world data. The use of real-world data in the simulator mitigates critical deficiencies inherent in offline RL agents [13]. Secondly, we assess our results using two evaluation metrics: Hit Ratio and S-Recall [15]. Specifically, in the context of Hit Ratio, we simulate multiple slates for test users. This simulation is done based on the simulation environment designed in Section 4. We then evaluate whether the actual items selected by the test user during the session are present within our simulated slates. In this way we are using real hold-out data for a stringent evaluation but also using the simulator to allow us to explore future long-term performance rather than one-shot performance. Both of the above points mitigates the concerns stated in [13] for state-of-the-art RL based slate recommendation.

Three key contributions are made. Firstly, we incorporate the user's diversity quotient in the RL reward function, enabling the system to learn to tailor news recommendations depending on the user's intent. Secondly, we address the scalability challenge of real-world RL deployments, by learning a two-phase policy function that, at inference, can cheaply select a set of candidate items on which to evaluate their more expensive network Q-function. Thirdly, we contribute a training process that combines a simulation environment with real-world data. This approach allows testing the agent on relevant metrics such as hit ratio on test data, enhancing result reliability.

2. Problem formulation

Our goal is to learn an agent policy to recommend a slate of items within which the user will choose one that matches their preferences (e.g. a list of YouTube videos from which the user selects one to watch). After the user consumes an item, they may choose to either receive additional slate recommendations or terminate the session. At each step t of the session, the RS recommends a slate $A_t = (i_t^1, \ldots, i_t^N)$, where $(i_t^j)_{1 \leq j \leq N}$ are items chosen among the candidate set of items $D_t \subseteq \mathcal{I}$, available at step t where $\mathcal{I}$ is the catalogue of items of the system and N is the number of items in the slate. To account for the possibility that the user may reject all items in a recommended slate, following [10], we include a null item denoted by $\perp$ in the $(N+1)^{th}$ position of each slate. To quantify user engagement, we consider the user's response when presented with a slate, which we treat as an observation received by the RS. The presented problem setting can be modelled as a Markov Decision Process (MDP), represented by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$, with γ indicating the discount factor, where $\mathcal{S}$ is set of all possible states representing a user, $\mathcal{A}$, set of all possible slates where $A \subseteq \mathcal{I}$ and $|A| = N$ is the slate size. $\mathcal{T}(s', s, A) = P(s'|s, A)$ a probability distribution over the next states s' given the current state s and the action A, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a reward function that maps each state-action pair to a real-valued reward representing the user satisfaction with the recommended slate.

A policy $\pi : \mathcal{S} \to \mathcal{A}$ dictates the action to be taken at any state. Denote is value function as V^π and action-value function as Q^π. The optimal policy π^* maximizes the expected reward over time and the optimal action-value function, Q^* is given by the fixed point of the Bellman equation:

$$Q^*(s, A) = R(s, A) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, A) V^*(s').$$

In [10], the combinatorially large action space of slate recommendation is addressed in the SlateQ system by decomposing the action value of a slate into action values of the items of which it is composed. This decomposition relies on having access to a *user choice model*, which quantifies the probability $P(d \mid s, A)$ of selecting an item $d \in A$ from a slate A in user state s, which we discuss later in the paper.

SlateQ depends on two underlying assumptions, that the user consumes only a single item from each slate and that the reward and transition depends only on the consumed item. These assumptions are reasonable on the basis that the user's engagement is not influenced to a great degree by the options in the slate that are not selected. The action value $Q^\pi(s, A)$ of a recommendation policy π, decomposes into item-wise action values $\overline{Q}^\pi(s, d)$ as follows:

$$Q^\pi(s, A) = \sum_{d \in A} P(d \mid s, A) \overline{Q}^\pi(s, d) \tag{1}$$

$$\overline{Q}^\pi(s, d) = R(s, d) + \gamma \sum_{s' \in S} P(s' \mid s, d) V^\pi(s') \tag{2}$$

where $R(s, d)$ denotes the reward when a user in state s consumes an item d. In this scenario, the authors show that Q-learning can proceed by applying temporal differencing on the item-wise Q-values. Off-policy learning requires successive solving of the optimisation problem

$$A \leftarrow \underset{\substack{A \subseteq I_c \\ |A| = N}}{\arg\max} \sum_{d \in A} P(d \mid s, A) \overline{Q}(s, d)\,. \tag{3}$$

3. Proto-Slate Framework

We combine SlateQ learning with the Wolpertinger policy [16], which was designed for very large discrete action spaces. Considering that items in our system have a feature vector representation $\mathbf{a} \in \mathbb{R}^m$, where m is the number of features, we can reason over actions in the *continuous* space $\mathbb{R}^{m \times N}$, where N is the size of the slate, which are dubbed *proto-actions* in [16], and which we will refer to as a *proto-slate*. In particular, a learning algorithm for continuous-valued actions can be employed to learn the optimal *proto-slate*. By firstly selecting a proto-action, the learning agent can then narrow down the set of potential discrete actions, in this particular case the set of items for constructing the slate, to a subset that can be retrieved based on the selected proto-action.

3.1. Learning slate representation

We define a parameterised function f_ϕ that maps a state s to a *proto-slate* as follows:

$$f_\phi : \mathcal{S} \to \mathbb{R}^{m \times N}, \;\; f_\phi(s) = \hat{\mathbf{A}}$$

where, $\hat{\mathbf{A}}$ is a real-valued matrix with m rows and N columns, representing the *proto-slate* corresponding to state s. Each column in the matrix represents an individual item representation in the continuous space. By defining a user choice model and reward function that generalizes to real-valued slates, Eq. 1 now represents the Q-value of a policy over the extended continuous action space. The intention is to learn the parameters ϕ so that f_ϕ converges to the optimal policy function over this action space of *proto-slates*.

Next, we define the function g_k to map the *proto-slate* $\hat{\mathbf{A}}$ to a set of k-nearest neighbor items in the feature space. The selection is performed independently for each item representation, resulting in a total of at most k candidate items, g_k is defined as:

$$g_k : \mathbb{R}^{m \times N} \to 2^{\mathcal{I}},$$
$$g_k(\hat{\mathbf{A}}) = \bigcup_{j=1}^{N} \Big(\text{nearest}(\hat{\mathbf{A}}[:, j], \|\cdot\|_2, \lfloor k/N \rfloor) \Big)$$

The candidate size k is split evenly between the N items, and the union operation $\bigcup$ merges the candidate items from all the individual representations, resulting in a set of at most k candidate items.

Learning the parameterized function f_ϕ to create a *proto-slate* representation and subsequently selecting $\lfloor k/N \rfloor$-nearest neighbor items for each individual representation through the function g_k, we aim to construct a diverse set of k candidate items pertaining to the various user profiles as discussed later. Being close to the *proto-slate*, these slates are expected to have high Q-values, so that coupled with a reward that takes diversity into account, this approach can facilitate effective action selections of diverse slates. We learn to maximize a policy that satisfies $\pi^*(s) = \arg\max_{A \subseteq g_k \circ f_\phi(s)} Q_\theta(s, A)$, where θ are the parameters of the Q network. Actions stored in the replay buffer are generated by policy π, but the policy gradient $\nabla_A Q(s, A)$ is taken at $\hat{\mathbf{A}} = f_\phi(s)$, where $Q(s, \hat{\mathbf{A}})$ is the Q-value of the proto-slate as computed using Eq.1.

4. Experimental Setup

To assess policies against the publicly available Mind dataset [17], we employ the RecSim [14] environment within the PyTorch framework. This dataset includes user click history, the news articles shown in the current session, and binary indicators indicating whether the user clicked on the articles (impressions). To align the dataset with the assumptions of the SlateQ algorithm, we replicate impressions such that instances featuring multiple clicks are disaggregated into individual clicks, so that for each interaction with a slate of news item during the session the user response is a single selection of an article.

User Model. The full version of the Mind (MIND-large) dataset [17] consists of 160k news articles with a million users and a disaggregated total of 15 million impression logs. We split the dataset into three parts. The first consists of all 639k users who have only a single impression. We use this subset to train the user-choice model. The remaining data is split into a train and test set. The test set consists of 30k randomly selected user sessions and is reserved for the final evaluation phase, while the training set is used to train the RL algorithm.

We leverage a non-sequential user modeling architecture, as described in [18], to learn click-through probabilities that then form the user-choice model. Specifically, we learn a function $h_\sigma(u, d)$ that maps a user u and item d to a click probability. Then given a slate A, the user choice model $P(d|u, A)$, is $h_\sigma(u, d)$ normalised, using softmax, over the items on the slate. As input to $h_\sigma()$, an item is represented by the GloVe embeddings [19] for the corresponding news article and a user by the GloVe embedding derived from the first news article in the user's history. The model is trained using binary responses y_{ud} of the user to items in the impression; it consists of dense layers and employs binary cross-entropy to measure the disparity between predicted probabilities and binary-valued responses.

Simulation Environment. We adopt 50-dimensional pre-trained GloVe word embeddings [19] for initialization of each of the news articles. The average over all of the embeddings of the user's clicked history articles is considered as the initial user's observed state represented as u_o. The candidate documents for each user are a set of 300 news articles which consist of the articles present in the user's impression in addition with randomly sampled articles from the dataset to make the total candidates for each user upto 300 denoted as D. To determine if a user is a generalist or specialist [20], we employ categorical entropy, a well-established information-theoretic measure that captures the uncertainty associated with the distribution of topics in a user's news history. This approach offers a principled way to assess user engagement breadth, as higher entropy d_u, signifies a more diverse range of interests and lower entropy indicates a specialization in specific topic categories [21].

Within the training set, the clicked item in each impression is represented as u_c. The hidden user state, indicative of the user's intent during a specific session and concealed from the agent, is computed as the average of all clicked news article embeddings for a given user u at the same timestamp t, denoted as u_h. In the reward model for a recommended slate A, the clicked article a_c is stochastically simulated using the trained user model. The reward comprises two components: the relevance of the selected document a_c with u_c, denoted as r_c, and user satisfaction, represented by the cosine similarity between u_h and a_c, denoted as S_u. These components are aggregated in the reward formula as [22]: $R(u, a_c) = (1 - d_u) \times r_c + d_u \times (1 - S_u)$. Diversity serves as the aggregator, reflecting the belief

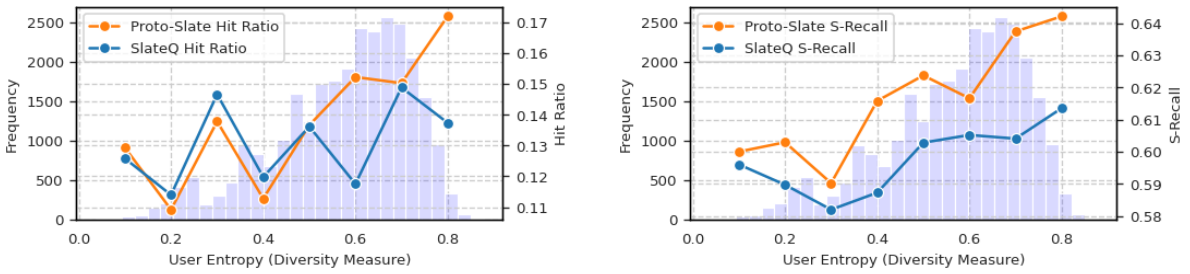
that the agent should be rewarded for selecting documents that diverge from the user's interest, while still ensuring relevance. The user's interests after consuming a news article are subject to stochastic nudge that bias them slightly towards an increase while also allowing for a chance of decrease. The adjustment's magnitude for the user state update is as follows: $u_o = u_o \pm (1 - G_{u,a_c}) \times a_c$, where the polarity is selected stochastically and G is the gaussian similarity as the interest of a user towards an article conforms to an inverted-U shape [23]. Therefore we design an *Undisclosed env.* for our agent that is it considers the environment a black-box and hence practically applicable.

Baselines. We evaluate our proposed method against five other baselines derived from previous work [10]. **Random**: This baseline generates a slate by randomly selecting items from the candidate set. **Greedy**: This baseline constructs a slate based on the scores predicted by the user choice model, selecting items with the highest scores until the slate is full. **SlateQ**: This serves as the state-of-the-art benchmark, employing a reinforcement learning approach for slate recommendation. To assess the effectiveness of learning a function that maps the state to a slate representation and subsequently reduces the candidate items for Q-function evaluation during inference, we compare Proto-Slate framework with two variants, **Random+SlateQ**: This variant employs SlateQ, but the candidate items are a random subset of the original candidate item set and **KNN+SlateQ**: This variant utilizes SlateQ with a subset of candidate items that are nearest neighbors to the state observed by the agent based on euclidean distance measure.

5. Experiments

We aim to analyze the presence of news article in the slate constructed by each algorithm as compared against the actual time-ordered clicked document of each test user during a session in a modified version of *Hit Ratio* as described in Section 1. Also we are interested in the proportion of number of topics present in the slate known as *S-Recall* for slate size of the recommended slates $N = 10$. Each training strategy is evaluated over $80,000$ user sessions that corresponds to $400K$ steps for the RL agent. Finally, each method tested against 300 user session trajectories. For all our runs we use the Adam optimizer with learning rates 0.0001 for both Q-networks and policy networks and a polyak averaging parameter [24], $\tau = 0.0001$. We set $\gamma = 1.0$ to check whether our policy architecture performs in the extreme non-myopic setting. To obtain more reliable results, we conducted 5 seeded runs of the experiment, and we report the mean and standard error of all metrics on the test data. The code for reproducibility is available on GitHub[1] and pseudocode is shared in the supplementary material.

Evaluation. Figure 1 shows the test data distribution of user diversity scores from 30,000 user



(a) Hit Ratio

(b) S-Recall

Figure 1: S-Recall and Hit Ratio by User Diversity against SlateQ and Proto-Slate.

sessions. Higher entropy indicates broader interests, while lower scores show more focused interests. The data is split into specialists (first quartile, entropy < 0.47; 7,000 sessions), generalists (entropy > 0.47; 21,500 sessions), and 1,500 cold-start sessions with no prior history. This stratification ensures a representative sample of user behaviors. To evaluate the algorithms using hit ratio and S-recall, we utilize our simulation environment. This environment generates multiple slates corresponding to the

[1]https://github.com/Asr419/rl_mind_dataset/

session length for each user in the test set by simulating item clicks through the learned user model and updating their state accordingly. We then calculate the hit ratio by checking if the actual items clicked during the session in the time-ordered test set is present in the corresponding slate generated during the session.

Table 1
Comparison of evaluation metrics for different slate recommendation strategies across different user profiles. The presence of the symbol † indicates that there is a statistically significant variation between the performance of a strategy and SlateQ, as identified by a paired t-test with p-value ≤ 0.05.

User Type →	Generalist		Specialist		Cold Start	
Strategy ↓	**Hit Ratio**	**S-Recall**	**Hit Ratio**	**S-Recall**	**Hit Ratio**	**S-Recall**
SlateQ	0.157 ± 0.013	0.604 ± 0.010	0.135 ± 0.011	0.587 ± 0.005	0.125 ± 0.010	0.599 ± 0.006
Random	$0.101 \pm 0.008^{\dagger}$	$0.535 \pm 0.004^{\dagger}$	$0.086 \pm 0.007^{\dagger}$	$0.531 \pm 0.005^{\dagger}$	$0.082 \pm 0.016^{\dagger}$	$0.531 \pm 0.009^{\dagger}$
Greedy	$0.121 \pm 0.008^{\dagger}$	$0.548 \pm 0.006^{\dagger}$	$0.115 \pm 0.016^{\dagger}$	$0.533 \pm 0.010^{\dagger}$	$0.110 \pm 0.007^{\dagger}$	$0.531 \pm 0.002^{\dagger}$
Random+SlateQ	$0.118 \pm 0.003^{\dagger}$	0.605 ± 0.006	$0.094 \pm 0.004^{\dagger}$	$0.605 \pm 0.007^{\dagger}$	$0.092 \pm 0.004^{\dagger}$	0.591 ± 0.009
KNN+SlateQ	$0.130 \pm 0.007^{\dagger}$	0.587 ± 0.008	0.126 ± 0.007	$0.577 \pm 0.005^{\dagger}$	0.121 ± 0.008	$0.582 \pm 0.004^{\dagger}$
Proto-Slate	0.154 ± 0.016	$0.630 \pm 0.008^{\dagger}$	0.132 ± 0.006	$0.603 \pm 0.007^{\dagger}$	0.129 ± 0.013	0.605 ± 0.009

In Table 1 we report result of Proto-Slate in comparison to other baselines. For all the candidate selection baselines and even Proto-Slate policy we report results for $k\%$ of candidate items where $k = 30$, although $k \in \{20, 30, 40\}$ had been tried on a single seeded run and $k >= 30$ for Proto-Slate gave statistically non-significant or better results across the metrics in comparison to SlateQ. This reduction in candidate subset enables reduction in the inference time in comparison to SlateQ policy. Although SlateQ is statistically significant when compared to other baselines except for Proto-Slate which is on the other hand significantly better in the topic coverage for generalist users. The advantage of learning a slate representation rather than getting the nearest items to the user state as done in our baseline for KNN+SlateQ is evident in the hit ratio for generalist users as it is outperformed by Proto-Slate. Each of the 5 seeded trained model runs is tested on the same 300 user session trajectory and are confirmed to be normally distributed and have equal variances respectively by the Shapiro-Wilk and Levene tests with a 95% confidence level making it appropriate for the conducted paired student *t-tests* for statistical analysis.

In Figure 1, we plot the hit ratio and S-recall for both SlateQ and Proto-Slate across different user entropy values. While the hit ratio is comparable for specialist users with both algorithms, Proto-Slate outperforms SlateQ in terms of slate diversity for generalist users. Proto-Slate's f-network learns to add items to the candidate set according to users' diversity preferences while maximizing the Q-value, resulting in slates with a comparable hit ratio to SlateQ. A significant advantage of Proto-Slate is its ability to curate slates with a greater number of topics (S-recall) for generalist users. To assess serving time efficiency, we compute the average time taken to serve a slate for each algorithm. The average inference time to serve a slate using SlateQ with a candidate size of 300 is **0.142s**, while Proto-Slate, with only 30% of the candidates, achieves comparable performance and user-specific diversity with an average serving time of **0.038s**.

6. Conclusion

In this study, we developed a real-world data-based simulation environment for slate recommendation using Reinforcement Learning (RL), enabling the recommendation of unseen or unlogged items in the existing dataset. We evaluate our simulation's performance based on actual clicks during test sessions. The proposed Proto-Slate policy shows promise in reducing serving time while achieving comparable performance and curating slates according to user diversity dynamics, compared to the

SlateQ algorithm. Proto-Slate excels for generalist users in terms of news topic diversity while maintaining performance for other user profiles.

Acknowledgments

This work was supported by the Science Foundation Ireland through the Insight Centre for Data Analytics under grant number SFI/12/RC/2289_P2.

References

[1] G. Bénédict, D. Odijk, M. de Rijke, Intent-satisfaction modeling: From music to video streaming, ACM Transactions on Recommender Systems 1 (2023) 1–23.
[2] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, M. Lalmas, Algorithmic effects on the diversity of consumption on spotify, in: Proceedings of the web conference 2020, 2020, pp. 2155–2165.
[3] N. Su, J. He, Y. Liu, M. Zhang, S. Ma, User intent, behaviour, and perceived satisfaction in product search, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 547–555.
[4] A. S. Roy, E. D'Amico, A. Lawlor, N. Hurley, Addressing fast changing fashion trends in multi-stage recommender systems, in: The International FLAIRS Conference Proceedings, volume 36, 2023.
[5] R. Mehrotra, M. Lalmas, D. Kenney, T. Lim-Meng, G. Hashemian, Jointly leveraging intent and interaction signals to predict user satisfaction with slate recommendations, in: The World Wide Web Conference, 2019, pp. 1256–1267.
[6] M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, E. H. Chi, Top-k off-policy correction for a reinforce recommender system, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, 2019, pp. 456–464.
[7] X. Chen, S. Li, H. Li, S. Jiang, Y. Qi, L. Song, Generative adversarial user model for reinforcement learning based recommendation system, in: International Conference on Machine Learning, PMLR, 2019, pp. 1052–1061.
[8] A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, I. Zitouni, Off-policy evaluation for slate recommendation, Advances in Neural Information Processing Systems 30 (2017).
[9] R. Deffayet, T. Thonet, J.-M. Renders, M. de Rijke, Generative slate recommendation with reinforcement learning (2023).
[10] E. Ie, V. Jain, J. Wang, S. Narvekar, R. Agarwal, R. Wu, H.-T. Cheng, T. Chandra, C. Boutilier, Slateq: A tractable decomposition for reinforcement learning with recommendation sets (2019).
[11] M. M. Afsar, T. Crump, B. Far, Reinforcement learning based recommender systems: A survey, ACM Computing Surveys 55 (2022) 1–38.
[12] A. Singha Roy, E. D'Amico, E. Tragos, A. Lawlor, N. Hurley, Scalable deep q-learning for session-based slate recommendation, in: Proceedings of the 17th ACM Conference on Recommender Systems, 2023, pp. 877–882.
[13] R. Deffayet, T. Thonet, J.-M. Renders, M. De Rijke, Offline evaluation for reinforcement learning-based recommendation: a critical issue and some alternatives, in: ACM SIGIR Forum, volume 56, ACM New York, NY, USA, 2023, pp. 1–14.
[14] E. Ie, C.-w. Hsu, M. Mladenov, V. Jain, S. Narvekar, J. Wang, R. Wu, C. Boutilier, Recsim: A configurable simulation platform for recommender systems, arXiv preprint arXiv:1909.04847 (2019).
[15] C. Zhai, W. W. Cohen, J. Lafferty, Beyond independent relevance: methods and evaluation metrics for subtopic retrieval, in: Acm sigir forum, volume 49, ACM New York, NY, USA, 2015, pp. 2–9.
[16] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, B. Coppin, Deep reinforcement learning in large discrete action spaces, arXiv preprint arXiv:1512.07679 (2015).

[17] F. Wu, Y. Qiao, J.-H. Chen, C. Wu, T. Qi, J. Lian, D. Liu, X. Xie, J. Gao, W. Wu, et al., Mind: A large-scale dataset for news recommendation, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 3597–3606.
[18] F. Tomasi, J. Cauteruccio, S. Kanoria, K. Ciosek, M. Rinaldi, Z. Dai, Automatic music playlist generation via simulation-based reinforcement learning, in: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2023, pp. 4948–4957.
[19] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
[20] I. Waller, A. Anderson, Generalists and specialists: Using community embeddings to quantify activity diversity in online platforms, in: The World Wide Web Conference, 2019, pp. 1954–1964.
[21] F. Eskandanian, B. Mobasher, R. Burke, A clustering approach for personalizing diversity in collaborative recommender systems, in: Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, 2017, pp. 280–284.
[22] S. Vargas, P. Castells, D. Vallet, Intent-oriented diversity in recommender systems, in: Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, 2011, pp. 1211–1212.
[23] B. Sguerra, V.-A. Tran, R. Hennequin, Discovery dynamics: Leveraging repeated exposure for user and music characterization, in: Proceedings of the 16th ACM Conference on Recommender Systems, 2022, pp. 556–561.
[24] B. T. Polyak, A. B. Juditsky, Acceleration of stochastic approximation by averaging, SIAM journal on control and optimization 30 (1992) 838–855.