A Good State Estimator Can Yield A Simple Recommender: A Reinforcement Learning Perspective

Dilina Chandika Rajapakse
rajapakd@tcd.ie
Trinity College Dublin
Ireland

Douglas Leith
doug.leith@tcd.ie
Trinity College Dublin
Ireland

ABSTRACT

In our study, we look at the application of (1) two offline Reinforcement Learning based recommenders (Decision Transformer and Prompt-based Reinforcement Learning PRL) and (2) a much simpler Neural Network based Value Network (MLP). We evaluate their performance in cold start conditions, where a user's preferences are not fully known, making recommendations challenging with uncertain 'states'. We show that in our experiments, the simple MLP value network outperforms both the Decision Transformer and PRL as well as Monte-Carlo Tree Search, the latter having previously shown state of the art performance in user-cold start recommendation. We also benchmark the performance of the MLP and transformer-based approaches under various conditions. We speculate that the 'state' estimation plays a key role in Reinforcement Learning based Recommendation Systems. With a good state estimation technique, even a basic Neural Network can be employed for effective recommendations, while requiring minimal computational power.

CCS CONCEPTS

• Information systems → Recommender systems; **• Computing methodologies →** *Reinforcement learning*.

KEYWORDS

Recommender Systems, Deep Learning, Offline Reinforcement Learning

ACM Reference Format:
Dilina Chandika Rajapakse and Douglas Leith. 2024. A Good State Estimator Can Yield A Simple Recommender: A Reinforcement Learning Perspective. In . ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3627673.3679083

1 INTRODUCTION

Personalisation, where the system caters to each user's individual preferences, is a key aspect of recommender systems. This is perhaps most conspicuously seen in user-cold start, where a recommender needs to quickly learn about a new user's preferences. To assist with this most systems leverage information about the existing users. One such approach is to cluster users into groups based on existing user data and then to select the items that a new user is asked to rate so as to quickly learn which group the new user belongs to.

Recent work [15, 16, 19] has shown how bandits and Monte Carlo Tree Search (MCTS) can be used to tackle this cold start task. MCTS treats the cold start task as a single-player game and aims to maximise a reward, e.g. the probability that the user is assigned to the correct group and/or the sum of the user's item ratings. MCTS maintains a user state that corresponds to the probability that the user belongs to each of the possible user groups and carries out a lookahead search for the next item to recommend. When run for sufficiently many steps it is guaranteed to find an optimal solution but a major practical limitation of MCTS is the computational cost of the exploratory search step, which increases exponentially with the size of the set of possible actions. Recently, it has been shown that this search step can sometimes be replaced by a transformer neural network trained in a supervised manner e.g. [18] show this for chess play.

In a parallel line of work, there has been a growing interest in using Reinforcement Learning (RL) for Recommender Systems, including transformer-based methods [23, 28]. This work has mostly been focused on generating item sequences given a history of user (item, rating, state) triplets. During cold start only a short history is available (or even none at all). In the literature the state is either assumed to be directly observed or is estimated in an ad hoc manner, although [10, 25] observe that state-estimation can have a substantial impact on recommender performance.

A common feature of these MCTS and RL approaches is that they effectively take the state as a given and mainly focus effort on the mapping from (item, rating, state) triples to the next item recommendation. In this paper we refocus attention on the state estimation aspect of these approaches. We examine cold start as the user's state is quickly evolving, providing a challenging test of state estimation. Our experimental results show that, given the state of a user, even a simple model such as an MLP achieves competitive recommendation performance. There no need for the computationally expensive search step in MCTS and there is no need to use complex deep learning neural nets in RL. Our results therefore suggest that we should place much more emphasis on finding good ways of estimating the state in MCTS and RL for recommenders, while time spent on developing complex models and reward policies may yield only limited benefits.

2 RELATED WORK

For a relatively recent survey of solutions to user cold-start see [6, 7]. Passive approaches include recommending popular items,

OARS 2024, October 25, 2024, Boise, ID, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679083

use of item-based recommendation, transfer learning from another recommender system previously used by a user, and asking new users to rate a fixed list of items. Examples of early work on active learning include information gain through clustered neighbors (IGCN) which uses a decision tree with user clusters as leaves [17] and the decision-tree approach of [8]. More recently, the group-based approach of [2] is extended to use a decision-tree approach by [20]. In [29] a matrix factorization approach is proposed whereby a decision-tree is trained to map from item ratings to the latent feature vector for a user. Seep learning methods for cold start are considered by [13, 22]. Recently, in [15, 16] Monte Carlo Tree Search (MCTS) is shown to achieve state of the art performance for user cold start.

Reinforcement learning techniques has shown promising results in sequential and long-term recommendations, see [1] for a survey of RL in RS. Value-based methods such as Deep Q-Networks make use of deep learning models to predicts the Q-values of all actions given the current state, or Q-value of a given state-action pair. Following the introduction of the transformer architecture in [21] for text processing, there has been interest in the applications of transformer neural networks in non-text domains. The Trajectory Transformer [11] and Decision Transformer (DT) [4] are perhaps the first papers to apply transformers to reinforcement learning but there is now a quickly growing literature on transformer-based RL, e.g. see [14, 26, 27]. In the context of recommender systems, CDT4Rec [23] and DT4Rec [28] have very recently proposed transformer-based RL methods. Similarly, [25] adopts the prompt based style in Decision Transformers, by formulating the offline RL task in a supervised manner.

RL based recommenders such as in [23–25], have shown to surpass non-RL methods like GRU4Rec[9], SASRec[12], in sequential and session-based recommendation tasks. However it can be argued that these next-item-prediction evaluations conducted in offline settings may not optimally test the explore-exploit capabilities of the RL recommenders [5]. Additionally, there is very limited literature on RL in cold-start recommendation.

3 PRELIMINARIES

3.1 Problem Formulation

Item recommendation is a Reinforcement Learning task where the RL agent (the recommender) interacts with the environment (the user) by taking actions (recommending items) and observing the reward (feedback from the users). The RL task is as follows: given a sequence of (state, item, rating) triplets $\{(s_i, v_i, r_i), i = 0, \dots, t\}$ predict the next item v_{t+1} to display so as to maximise the sum-rating $R_t = \sum_{i=0}^{t} r_i$. The state is an information state that embodies user preferences and context. In the general RL literature the state is usually assumed to be directly observed, while in the recommender RL literature it is more commonly estimated e.g. using an LSTM.

3.2 State Estimation

In order to focus on the RL aspect of cold start recommendation, we construct our experiments so that the ground truth user state is known. There is a set $\mathcal{G}$ of user groups, each user belonging to one group $g \in \mathcal{G}$. For users belonging to group g the rating $R(v)$ of item v is i.i.d. gaussian with mean $\mu(g, v)$ and variance $\sigma^2(g, v)$. If asked repeatedly to rate the same item then the user responds with the same rating. Denoting the user's group by random variable G, we have that $p(R(v) = r|G = g) = (1/\sqrt{2\pi}\sigma(g, v))e^{-(r-\mu(g,v))^2/2\sigma^2(g,v)}$. We let $p_g^{(t)}$ be the probability that the user belongs to group g given the user has rated items $\mathcal{V}^{(t)} = \{v_1, \dots, v_t\}$. Initially, for a new user $t = 0$, $\mathcal{V}^{(0)}$ is the empty set and the probabilities $p_g^{(t)}, g \in \mathcal{G}$ are initialised to the uniform distribution $p_g^{(0)} = 1/|\mathcal{G}|$. The state of a user is the vector of group probabilities $P^{(t)} = (p_1^{(t)}, p_2^{(t)}, \cdots, p_{|\mathcal{G}|}^{(t)})$.

Incorporating trainable state-encoders, such as GRUs or CNNs, can be commonly seen in RL literature. These need to be trained concurrently with the RL-models. As a result of this, the state generated by the encoders will differ across models since the learned weights are specific to each training instance. We considered our group-membership based state to be a good state estimate to conduct our experiments in a controlled and reproducible manner.

3.3 Decision Transformer

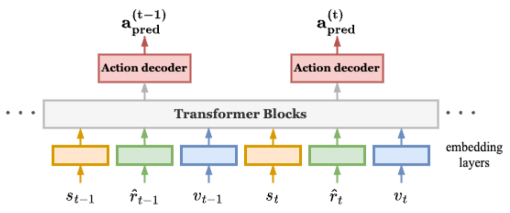

Figure 1: Decision Transformer Architecture

The Decision Transformer (DT) [4] uses a transformer neural net for Reinforcement Learning, see Figure 1. To generate the t'th item recommendation the input consists of a sequence of past (state, reward, item) triplets (s_i, r_i, v_i) for $i = 0, \dots, t-1$ plus $(s_t, \hat{r}_t)$ where $\hat{r}_t$ is the target rating for the next item (for a recommender we use the highest possible item rating). The Decision Transformer outputs a probability distribution over the set of items. We select the item with highest probability not yet rated by the user as the recommendation.

3.3.1 *Input Embedding.*

The inputs $\{(s_i, r_i, v_i,), i = 0, \dots, t\}$ are passed through an embedding layer, which maps the inputs into a sequence of high-dimensional token embeddings with a size $d = 128$. To embed the temporal information of the inputs, positional encodings are added to the corresponding input embeddings, before being fed into the transformer blocks.

3.3.2 *Transformer Block.*

The input token embeddings are fed to a stack of N transformer blocks, which consist of multi-head self attention layers. A causal attention mask ensures that only the prior inputs are attended. We use 2 transformer blocks ($N = 2$) with 4 heads ($h = 4$) in each attention layer.

3.3.3 *Action Decoder.*

The output from the transformer blocks is passed through an action decoder, which consist of a linear layer, followed by a softmax activation.

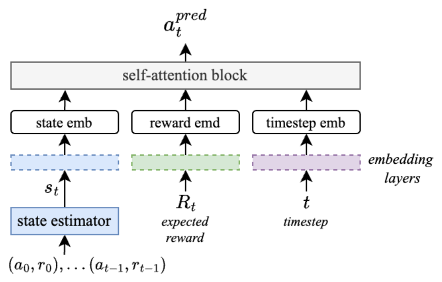

Figure 2: PRL recommender Architecture

3.3.4 *Loss and Training.* The training data consists of sequences of (item,rating) pairs for a population of users. The items are selected randomly. We use the cross entropy loss between the decision transformer prediction and the item v_t in the training data.

3.4 Prompt-based Reinforcement Learning

Prompt-based Reinforcement Learning [25] (PRL) is an offline-RL approach specifically designed for recommender systems. Similarly to the Decision Transformer, the input is a history of (reward, item) pairs together with the target reward for the next item. The output is a probability distribution over the set of items and we select the item with highest probability not yet rated by the user as the recommendation. The architecture is shown schematically in Figure 2.

PRL is trained in a supervised manner, similarly to the Decision Transformer. For each offline collected sequence of (item,reward) pairs with length t, a prompt is generated which includes the current state, reward and the timestep. In [25] the current state s_t is calculated using the past history of item interactions (v_i, r_i) for $i = 0, \dots, t-1$ using a variety of sequential models but here we use the state estimation method mentioned in Section 3.2. The state s_t, reward r_t and timestep t are mapped to latent representations using embedding layers. These are then passed into a self-attention block whose output is a probability distribution over the set of items. A cross-entropy loss is calculated between the predicted item vector $\mathbf{a}_\mathbf{t}^{\mathbf{pred}}$ against the observed item v_t in the training data.

During inference, the input prompt is generated using an expected reward R_t, along with the state s_t at timestep t. In our experiments, R_t was set to the highest possible item reward.

3.5 MLP Value Network

As a simpler alternative we also consider the MLP recommender shown in Figure 3.

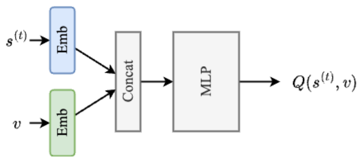

Figure 3: MLP Value Network Architecture

The input is a (state, item) pair (s^t, v). The output is the predicted rating for item v. We select the item with highest rating not yet rated by the user as the recommendation.

3.5.1 *Input Embedding.* The state s^t and item v inputs are mapped to d dimensional vectors via an embedding layer. We use an embedding size of $d = 128$.

3.5.2 *Value Estimator.* The input embeddings are passed through an MLP with one hidden layer having the number of neurons equal to the embedding size. The output is the predicted reward for the input item.

3.5.3 *Loss and Training.* The training data used is the same as that for the Decision Transformer. We use mean square error loss between the predicted rating and the item rating in the training sequence.

4 MEASURED PERFORMANCE

4.1 Experimental Setup

4.1.1 *Simulation Environment.* To allow us to evaluate the performance in a clean, reproducible manner we use a simulation environment that allows us to generate users with known ground-truth item-ratings. Similarly to [15, 16, 19], each simulation environment is derived from a measurement dataset consisting of (user, item, ratings) triples. We use three public datasets Netflix[1], Goodreads[2] and Movielens10M[3] . For each dataset we cluster users into groups (we use the BLC matrix-factorization clustering algorithm [3] for this, although other clustering algorithms (such as k-means) might also be used) and estimate the mean $\mu(g, v)$ and variance $\sigma(g, v)^2$ of the ratings by each group g for item v. For each user belonging to group g, we generate the rating for an item v by making a single draw from the multivariate Gaussian distribution with mean $\mu(g, v)$ and variance $\sigma(g, v)^2$.

This environment enables the generation of sparse user-rating data for training, allowing us to immitate a recommender system dataset. For each user in a group $g \in \mathcal{G}$, we randomly sample between 10 and 200 items. Although item sampling based on popularity (i.e., the number of interactions) is an option, we choose to sample items uniformly to reduce potential biases within the training data as far as possible. For evaluation, we generate user item-ratings for 250 test users from each user group.

It is also possible to allow the RL agent to interact with the simulation environment and follow an on-policy learning strategy, commonly employed in RL applications. However this approach is often impractical in the context of recommender systems.

4.1.2 *Metrics. Mean rating across iterations $\overline{R}_{@t}$:* The mean item rating is $\overline{R}_{@t} = \frac{1}{t}\sum_{i=0}^{t-1} r_i$ where r_i is the rating of the i'th item presented to a user. This corresponds to the sum-rating utility that we would like to maximise in the RL setup, and so is our primacy performance measure.

[1]https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data
[2]https://mengtingwan.github.io/data/goodreads
[3]https://grouplens.org/datasets/movielens/

4.1.3 Baselines. We evaluate the cold-start performance of the Decision Transformer (DT), Prompt-based RL recommender (PRL) and the MLP Value network against two baselines:

(1) *Monte Carlo Tree Search (MCTS)*: This is a strong baseline that represents state of the art user cold start performance. We use a variant of the MCTS[16] recommender, that takes an RL approach to not only focus on learning the user-groups, but also maximising a sum rating reward.
(2) *Random-Uniform (R-U)*: This presents items selected uniformly at random from the items not yet viewed by the user. This random baseline represents the population of training data, on which our DT and MLP are trained on.

4.1.4 Hardware and Software. The training and evaluations were conducted on a machine with a 32-core AMD Ryzen CPU and 2 × NVIDIA GeForce RTX 4090 GPUs. Our implementation[4] also includes the recommendation environment and the datasets.

[4]https://github.com/dilina-r/rl_estimator

dataset	*algo*	*iterations*				
		5	*10*	*15*	*20*	*25*
Netflix 8	R-U	3.231	3.219	3.214	3.211	3.213
	MCTS	3.837	3.968	4.019	4.050	4.069
	PRL	4.546	4.475	4.430	4.409	4.387
	DT	4.559	4.487	4.445	4.422	4.407
	MLP	**4.566**	**4.498**	**4.463**	**4.434**	**4.417**
Goodreads 8	R-U	3.535	3.542	3.546	3.545	3.543
	MCTS	3.839	3.882	3.866	3.858	3.854
	PRL	4.140	4.148	4.130	4.103	4.082
	DT	4.260	4.218	4.189	4.164	4.146
	MLP	**4.268**	**4.237**	**4.204**	**4.173**	**4.149**
Movielens 8	R-U	3.166	3.157	3.161	3.160	3.159
	MCTS	3.753	3.814	3.846	3.867	3.880
	PRL	4.096	4.083	4.082	4.068	4.062
	DT	4.179	4.149	4.135	4.119	**4.109**
	MLP	**4.181**	**4.152**	**4.138**	**4.122**	4.106
Netflix 16	R-U	3.319	3.325	3.322	3.321	3.321
	MCTS	3.773	3.883	3.942	3.978	4.006
	PRL	4.465	4.404	4.352	4.344	4.327
	DT	**4.532**	4.437	4.393	4.373	4.362
	MLP	4.531	**4.451**	**4.407**	**4.385**	**4.368**
Goodreads 16	R-U	3.604	3.604	3.606	3.609	3.609
	MCTS	3.815	3.855	3.877	3.886	3.893
	PRL	4.189	4.120	4.087	4.051	4.013
	DT	**4.241**	4.188	4.148	**4.121**	**4.104**
	MLP	4.240	**4.194**	**4.149**	4.118	4.091
Movielens 16	R-U	3.360	3.360	3.369	3.368	3.367
	MCTS	3.849	3.893	3.920	3.940	3.950
	PRL	4.153	4.089	4.046	4.027	4.017
	DT	4.191	**4.172**	**4.166**	**4.162**	**4.154**
	MLP	**4.194**	4.168	4.148	4.134	4.121

Table 1: Mean ratings across t recommendations ($\overline{R}_{@t}$) with Random-uniform (R-U), MCTS, PRL, Decision Transformer (DT) and Value Network (MLP) for Netflix, Goodreads, Movielens10M datasets with 8 and 16 groups

4.2 MLP Outperforms Other Methods

When a new user joins the system, the system initially presents an item to the user and in turn the user provides a feedback (i.e: rating). The system then continues to present the user with items, while learning from the feedback to the previously recommended items. To evaluate the cold-start performance, we generate new users belonging to each group $g \in G$, from the simulation environment.

Table 1 shows the mean reward for between 5 and 25 item recommendations measured for the Netflix, Goodreads and Movielens10M datasets with 8 and 16 groups. The highest values are indicated in bold. Note that this is a fair comparison in the sense that all of the approaches have access to the same user data, including the same state estimate.

Surprisingly, at least to the authors, it can be seen that the MLP value network consistently achieves the better performance compared to PRL and MCTS. The Decision Transformer and MLP are on par, each showing superior performance under different datasets and conditions. This is despite the fact that the MLP is considerably simpler than the Decision Transformer and PRL architectures. The MLP is also much simpler than the MCTS approach which requires an expensive exploratory lookahead search for the best item.

This has significant implications. In particular, it means that there is little value to be gained by adding complexity to the RL component of DT, PRL and MCTS recommenders. Indeed in our experience adding extra complexity may lead to a loss in performance. For the DT this loss in performance is presumably due to the greater difficulty of training, while for MCTS the high computational burden means that the exploration needs to be curtailed which can degrade performance.

By design, the state is explicitly known in these tests so as to avoid confounding effects associated with state estimation. We leave evaluation of the impact of state estimation errors to future work, but note that our results suggest that the primary effort in RL recommenders should be devoted to designing good state estimators. In contrast, most of the current recommender RL literature focuses on designing the mapping from (state, reward, item) triplets to the next item, which of course is also the main focus of the RL literature generally.

It is also worth noting that the Decision Transformer not only receives a state input at each step, but also receives the (item,reward) pairs of the user's past interactions. The history of past (item,reward) pairs is enough to allow the state to be calculated.

4.3 Computational Cost

In addition to achieving strong recommendation performance, the MLP value network is also cheaper computationally than the Decision Transformer and PRL. This can be seen from Table 2 which shows the measured compute time of the MLP, DT and PRL for the Netflix dataset as the size of the set of items available to be recommended is varied. The compute time shown is the average time to recommend 25 items to a user, including the state-estimation and model prediction times. Also shown in Table 2 is the GPU memory used by each approach during the evaluations.

This is perhaps unsurprising given the simplicity of the MLP, but recall that the MLP outputs the predicted rating for a single item and so to predict the next item to recommend it needs to be run for

dataset	*#items*	*no. of parameters*			avg time/25 recs (ms)			allocated GPU mem (MB)			$\overline{R}_{@25}$		
		MLP	*DT*	*PRL*	*MLP*	*DT*	*PRL*	*MLP*	*DT*	*PRL*	*MLP*	*DT*	*PRL*
Netflix-S	752	34,689	546,672	227,312	8.92	43.89	14.67	8.63	11.48	8.99	**4.417**	4.407	4.387
Netflix-M	4,804	34,689	1,069,380	750,020	11.60	43.01	14.86	10.60	16.62	10.99	4.374	**4.380**	4.326
Netflix-L	15,020	34,689	2,387,244	2,067,884	15.92	43.52	15.20	15.59	29.55	16.01	4.358	**4.361**	4.298

Table 2: Average recommendation time per user, model parameter count, GPU memory usage, Mean reward $\overline{R}_{@25}$ for the Value Network (MLP) and Decision Transformer (DT), with varying available number of items - Netflix 8 groups dataset

every item not yet rated by the user. Thus the computational cost of the MLP increases linearly with the number of items, although this might be compensated by implementing batch-predictions. The Decision Transformer and PRL both output a probability vector over the set of possible items, which scales with the number of items. However, it can be seen from Table 2 that the compute time for DT and PRL increases slightly with the number of items and so the compute burden is dominated by other components of these networks.

MCTS data is not included in Table 2 because we found that the computational burden quickly becomes too high as the number of available items is increased it grows exponentially in the number of items). We also note that MCTS is not well suited to GPU implementation and so is difficult to benchmark fairly against the MLP and DT approaches which use a GPU.

The last column of Table 2 also summarises how the recommendation performance varies with the number of available items. It can be seen that the performance tends to fall as the number of items is increased, presumably reflecting the increased difficulty of finding the best item to recommend when the pool of available items is larger. However, it can also be seen that the decrease in performance is much more pronounced for PRL compared to the MLP and DT.

5 CONCLUSION

Our results suggest that there is little value to be gained by adding complexity to the RL component of DT, PRL and MCTS recommenders. Indeed the extra complexity may lead to a loss in performance. Rather the primary effort in RL recommenders should be devoted to designing good state estimators.

REFERENCES

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2022. Reinforcement learning based recommender systems: A survey. *Comput. Surveys* 55, 7 (2022), 1–38.

[2] Xavier Amatriain, Neal Lathia, Josep M. Pujol, Haewoon Kwak, and Nuria Oliver. 2009. The Wisdom of the Few: A Collaborative Filtering Approach Based on Expert Opinions from the Web. In *Proc SIGIR* (Boston, MA, USA). 532–539. https://doi.org/10.1145/1571941.1572033

[3] Alessandro Checco, Giuseppe Bianchi, and Douglas J. Leith. 2017. BLC: Private Matrix Factorization Recommenders via Automatic Group Learning. *ACM Trans. Priv. Secur.* 20, 2, Article 4 (May 2017), 25 pages. https://doi.org/10.1145/3041760

[4] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 15084–15097. https://proceedings.neurips.cc/paper_files/paper/2021/file/7f489f642a0ddb10272b5c31057f0663-Paper.pdf

[5] Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten De Rijke. 2023. Offline evaluation for reinforcement learning-based recommendation: a critical issue and some alternatives. In *ACM SIGIR Forum*, Vol. 56. ACM New York, NY, USA, 1–14.

[6] Mehdi Elahi, Matthias Braunhofer, Tural Gurbanov, and Francesco Ricci. 2018. User Preference Elicitation, Rating Sparsity and Cold Start. *Collaborative Recommendations* (2018), 253–294. https://doi.org/10.1142/9789813275355_0008

[7] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2016. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review* 20 (2016), 29–50. https://doi.org/10.1016/j.cosrev.2016.05.002

[8] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. 2011. Adaptive Bootstrapping of Recommender Systems Using Decision Trees. In *Proc WSDM* (Hong Kong, China). 595â604. https://doi.org/10.1145/1935826.1935910

[9] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[10] Jin Huang, Harrie Oosterhuis, Bunyamin Cetinkaya, Thijs Rood, and Maarten de Rijke. 2022. State encoders in reinforcement learning for recommendation: A reproducibility study. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2738–2748.

[11] Michael Janner, Qiyang Li, and Sergey Levine. 2021. Offline Reinforcement Learning as One Big Sequence Modeling Problem. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 1273–1286. https://proceedings.neurips.cc/paper_files/paper/2021/file/099fe6b0b444c23836c4a5d07346082b-Paper.pdf

[12] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. IEEE, 197–206.

[13] Sergio Oramas, Oriol Nieto, Mohamed Sordo, and Xavier Serra. 2017. A deep multimodal approach for cold-start music recommendation. In *Proceedings of the 2nd workshop on deep learning for recommender systems.* 32–37.

[14] Aaron L Putterman, Kevin Lu, Igor Mordatch, and Pieter Abbeel. 2021. Pretraining for language conditioned imitation with transformers. (2021).

[15] Dilina Chandika Rajapakse and Douglas Leith. [n. d.]. User Cold-Start Learning In Recommender Systems Using Monte Carlo Tree Search. *ACM Transactions on Recommender Systems* ([n. d.]).

[16] Dilina Chandika Rajapakse and Douglas Leith. 2022. Fast and Accurate User Cold-Start Learning Using Monte Carlo Tree Search. In *Proceedings of the 16th ACM Conference on Recommender Systems.* 350–359.

[17] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Explor. Newsl.* 10, 2 (Dec. 2008), 90â100. https://doi.org/10.1145/1540276.1540302

[18] Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. 2024. Grandmaster-level chess without search. *arXiv preprint arXiv:2402.04494* (2024).

[19] Sulthana Shams, Daron Anderson, and Douglas Leith. 2021. Cluster-based bandits: Fast cold-start for recommender system new users. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1613–1616.

[20] Lei Shi, Wayne Xin Zhao, and Yi-Dong Shen. 2017. Local Representative-Based Matrix Factorization for Cold-Start Recommendation. *ACM Trans. Inf. Syst.* 36, 2, Article 22 (Aug. 2017), 28 pages. https://doi.org/10.1145/3108148

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[22] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. *Advances in neural information processing systems* 30 (2017).

[23] Siyu Wang, Xiaocong Chen, Dietmar Jannach, and Lina Yao. 2023. Causal Decision Transformer for Recommender Systems via Offline Reinforcement Learning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Taipei</city>, <country>Taiwan</country>, </conf-loc>) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 1599–1608. https://doi.org/10.1145/3539618.3591648

[24] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. 2020. Self-Supervised Reinforcement Learning for Recommender Systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 931–940. https://doi.org/10.1145/3397271.3401147

[25] Xin Xin, Tiago Pimentel, Alexandros Karatzoglou, Pengjie Ren, Konstantina Christakopoulou, and Zhaochun Ren. 2022. Rethinking reinforcement learning for recommendation: A prompt perspective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1347–1357.

[26] Mengdi Xu, Yikang Shen, Shun Zhang, Yuchen Lu, Ding Zhao, Joshua Tenenbaum, and Chuang Gan. 2022. Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*. PMLR, 24631–24645.

[27] Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. 2023. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*. PMLR, 38989–39007.

[28] Kesen Zhao, Lixin Zou, Xiangyu Zhao, Maolin Wang, and Dawei Yin. 2023. User Retention-oriented Recommendation with Decision Transformer. In *Proceedings of the ACM Web Conference 2023*. 1141–1149.

[29] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional Matrix Factorizations for Cold-Start Recommendation. In *Proc SIGIR*. 315â324. https://doi.org/10.1145/2009916.2009961