Developing Sim-to-Real Multi-Task Recommendations via Open-Ended Learning

Anirudh Jamkhandi
ZOZO Inc.
Chiba, Japan
anirudh.jamkhandi@zozo.com

Masahiro Yasuda
ZOZO Inc.
Chiba, Japan
masahiro.yasuda@zozo.com

Shusaku Yosa
ZOZO Inc
Chiba, Japan
shusaku.yosa@zozo.com

ABSTRACT

Collaborative Interactive Recommenders (CIRs) have received growing attention due to their capability to model sequential user interactions and optimize users' long-term engagement and overall satisfaction. Although the long-term planning perspective of CIRs provides a natural setting for using reinforcement learning (RL), RL methods require a large amount of online user interaction, which is restricted due to economic considerations. Previous research has concentrated on developing simulators to simulate user actions in real-world systems, intending to maximize only one type of user response. But real-world recommenders have to simultaneously solve multiple recommendation tasks, each of which is centered on optimizing a single type of user feedback, making the sim-to-real transfer challenging. This is further exacerbated by the severe partial observability and stochasticity of real-world recommendation environments.

To address the above issues, we propose a method that employs an open-ended learning algorithm to address the long-term rewards of recommendations while handling multiple recommendation tasks at the same time. This solution amalgamates the fundamental concepts of goal switching, novelty search, and quality diversity into an open-ended recommender agent that can create complex and diverse recommendation environments and simultaneously learn to rank on them. With enough diversity in the simulator, the real world may appear to the model as just another variation. We conduct a comprehensive evaluation on real large-scale data containing millions of events and demonstrate that our solution outperforms many baselines on standard evaluation metrics.

KEYWORDS

multi-task learning, Open-Ended Learning, Deep Reinforcement Learning

1 INTRODUCTION

With the explosion of online content, users are often bombarded with innumerable content. Recommendation systems are a valuable tool to reduce information overload, increase stakeholder revenue, and improve user experience [6]. Recommendation techniques are now widely studied and used in numerous applications.

Traditional recommendation techniques [2, 4, 10, 11, 18, 36, 42, 45, 46, 51, 52, 54] tend to focus on maximizing the immediate (short-term) rewards of current recommendations, while overlooking the long-term rewards that can be generated by current recommendations. Recently, Collaborative Interactive Recommenders systems (CIRs) have received growing attention due to its awareness of long-term engagement and dynamic preference. In CIRs, RL techniques have been recently gaining attention, showing their advantages in accommodating dynamic user preferences [8, 9, 48].

But previous research on RL techniques for CIRs has focused on a single recommendation task that aims to maximize a single type of user response, such as a click. But in many real-world applications, we are often confronted with multiple recommendation tasks, each of which requires improving on a single type of user feedback. Recommender systems (RS) need to incorporate various user feedbacks to model user interests and maximize user engagement and satisfaction. Therefore, it is desideratum to apply multi-task learning (MTL) in RS to simultaneously model the multiple aspects of user satisfaction or engagement. And in fact, it has been the mainstream approach in major industry applications [27, 29, 47]. In order to optimise multiple tasks, large amount of experience and computation is required to learn the desired policies. Hence to train agents with limited interaction data, existing RL models in CIRs often assume that the simulation task and the real task are the same recommendation task, and count on building simulators to mimic user behaviors in real systems. However, in practice the trained models can achieve poor performances due to poor generalization across users, noisy and dynamic observed behavior of user and exogenous unobservable events.

In this paper, we propose a method that utilizes open-ended learning to address the need for solving multiple recommendation tasks together, each of which corresponds to a single type of user feedback and maximizing long-term rewards. The open-ended agent models the interaction between the recommender and user by Markov decision process (MDP), taking into account of long-term impacts of current recommendation to subsequent rewards. The recommendation action incurs the state transition of the user, and the user's subsequent responses to later recommendations will be dependent on the transited state. The open-ended agent automatically generates a curriculum of recommendation environment of ever-increasing difficulty. If the variability in simulation is significant enough, models trained in simulation will generalize to the real world with no additional training. The open-ended learning agent employs a deep neural network (DNN) to learn state representation and the optimal policy for each recommendation task. To facilitate knowledge transfer and improve the performance over multiple tasks the agent learns a multi-task policy by combining multiple expert policies learned for different tasks into a single multi-task policy that can outperform the separate experts.

Our three main contributions are :

(1) We propose a novel multi-task recommendation framework based on open-ended learning, that endlessly generates different recommendation environments and learns to rank. This is the first solution to apply open-ended learning to recommendation tasks. This solution allows us to obtain a model with a smaller size and lower response latency, making it more appealing for real-world deployments.
(2) We also build a simulator based on Recsim Framework that enables us to design multi-task long-term reward recommendation environments. To our knowledge, this solution is the first of its kind in the simulator to real transfer recommendation systems.
(3) We conduct a comprehensive offline experimental evaluation over the ZOZOTOWN E-Commerce Platform, which is Japan's largest Fashion E-Commerce. The evaluation results on millions of real-world log events illustrate that our model achieves competitive performance over many state-of-the-art methods based on multiple evaluation metrics.

The rest of this paper unfolds as follows: In Section 2, we review related works. In Section 3, we provide problem descriptions. Next, we talk about our proposed approach. Section 5 elucidates how we design a long-term reward recommendation environment and experiments to evaluate our proposed framework. Finally, we draw our conclusion in Section 6.

2 RELATED WORKS

Reinforcement Learning. To address the long-term rewards, reinforcement learning (RL) has been applied to different recommendation tasks, including video recommendation [7, 8, 20, 26], e-commerce recommendation [9, 33, 44, 55, 56], news recommendation [58], and treatment recommendation [48]. Unlike conventional techniques, RL models consider that rewards of recommendation are state dependent, and that the current recommendation causes a state transition. The goal of RL models is to learn an agent's action policy to maximize expected long-term rewards in a series of interactions between the agent and the environment (e.g., the user) [40]. These RL-based recommenders outperform previous best practices in several evaluation metrics. However, because these recommenders are designed to optimize a single type of user feedback, they cannot handle multiple recommendation tasks. This paper focuses on tackling such real-world driven problem.

Multi-Task Learning. Multi-task Learning (MTL) models leverage useful information shared in multiple related tasks to help improve the performance of all the tasks [53]. The cross-stitch network [31] learns a unique combination of task-specific hidden-layer embeddings for each task. MOE [21] first proposes to share some experts at the bottom and combine experts through a gating network. The YouTube video recommender system in [57] , applies MMOE [28] to combine shared experts through different gating networks for each task. To address seesaw phenomenon commonly observed in MTL models PLE [41] explicitly separates task-common and task-specific experts and uses progressive separation routing. Hadash et al. [16] propose a multi-task framework to learn parameters of the ranking task and rating task simultaneously. The task of text recommendation in [3] is improved through sharing representations at the bottom. Jing et al. [22] simultaneously solve the user returning time prediction task and the recommendation task. Most of these prior works do not address multiple recommendation tasks simultaneously, let alone considering long-term rewards of recommendations.

Open-Ended Learning. Artificial life researchers have been studying the prospects of open-ended computation [5, 15, 24, 25, 39, 49, 50] since many years. Open-ended learning research is also reflected in recent advances in automatic curriculum learning for RL, where the intermediate goals of curricula are automatically generated and selected [1, 30]. Works such as in [14, 43] have also demonstrated the zero-shot generalization capability of open-ended learning. None of the methods have applied open-ended learning to recommendation tasks. We see this as an interesting direction for future work.

3 PROBLEM STATEMENT

Recommendation task can be described as a sequential decision process, in which the recommender decides a sequence of recommendations for the users. Tasks are specified by a Markov Decision Process (MDP), defined as tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, state space $\mathcal{S}$ represents user's and items characteristics and behavior (eg., interests, preferences, satisfaction, activity, mood, etc.) which can be both observable and latent, set of all items as action space $\mathcal{A}$, transition function $\mathcal{P}(\cdot|s,a)$, a feedback reward function $\mathcal{R}(s,a)$, decay factor γ and the recommender as the "agent". An action from the agent causes state transition of the environment, and consequently, the agent will make a decision at the next time step based on the transited state.

In order to optimise multiple tasks, one can train a separate policy for each task. However a large amount of experience and computation is required to learn the desired policies. It is therefore preferable to have a single agent that facilitates the exchange of information among various tasks. To effectively model the dependency among these types of feedback and objectives we reuse policies for different tasks by adopting shared state-space $\mathcal{S}$ and action-space $\mathcal{A}$ across tasks. The tasks only differ in their transition function P_τ and reward function R_τ. We therefore describe a task as $\tau = (\mathcal{P}_\tau, \mathcal{R}_\tau)$ and refer to the set of given tasks as T. For each task $\tau \in T$ we aim to maximize the discounted return $G_\tau = \sum_{t=0}^{t=L} \gamma^t r_\tau^t$ where $r_\tau^t \sim \mathcal{R}(s_t, a_t)$ is the reward at time step t and L is the episode length. Given a set of policies $\{\pi_1, ..., \pi_n\}$, we denote the return obtained by policy π_i on task π as $G_\tau(\pi_i)$.

4 PROPOSED SOLUTION

To this end we propose open-ended learning to integrate multi-task learning into RL based recommender. Although RL-based recommenders provide an excellent way to model long term dynamics of the users, the large action-space combinatorics prevent them from being employed in practical recommenders. We adapt Paired open-ended Trailbrazer(POET) algorithm as discussed in [49] for recommendation tasks. We reduce the action space to three continuous values(co-efficients of the ranking formula). This drastically reduces the number of computations and improves the latency. The POET recommender agent builds curricula of increasingly complex and diverse recommendation environments and simultaneously learns to optimize on multiple recommendation tasks. There are several reasons that make POET algorithm suitable for multi-task recommendations. Firstly, the challenge in black-box optimization is the presence of local optima, making it impossible for the search process to move on to better points [13]. Open-Ended Algorithms like POET use Novelty Search which focus on divergence instead of convergence. The offsprings are chosen solely on the basis of their behavioral divergence from an archive of individuals from previous generations. Truly novel discoveries are often building blocks to further novel discoveries. Sometimes, completely diverging from an objective can help optimize faster than pursuing it [23]. This allows the algorithm to discover complex challenges which may occur in the real world as well. Secondly, POET implements coevolutionary interaction among all its agents and environments and also optimizes the behavior of each agent within its paired environment. It ensures Quality Diversity by maintaining only those newly-generated environments that are not too hard and not too easy for the current population of agents. This assures that the algorithm not only discovers increasingly difficult challenges but also simultaneously learns to solve them. The above mentioned features enable simulator to real world adaptation [12]. Finally, it encourages knowledge sharing by periodically testing the performance of offspring agent from one environment in other environments through Goal Switching. This allows recommender's performance to improve on multiple tasks by solving them jointly.

We adopt the simplest model in reinforcement learning ie. Evolutionary Strategies (ES) to derive the optimal recommendation policy. In ES, $E(\psi)$ represents the stochastic reward experienced over a full episode of an agent interacting with the environment $E(\cdot)$. ES seeks to maximize the expected fitness over a population of parameterized policy whose parameter vector is denoted as ψ, $J(\theta) = \mathbb{E}_{\psi \sim p_{\theta(\psi)}}[E(\psi)]$, where ψ is sampled from a probability distribution $p_{\theta(\psi)}$ parameterized by θ.

Following the approach of Salimans et al. [38], the gradient of J(θ) with respect to θ can be written as

$$\nabla_\theta J(\theta) \approx \frac{1}{n\sigma} \sum_{i=1}^{n} E(\theta + \sigma\epsilon_i)\epsilon_i \tag{1}$$

where, n samples of parameter vectors drawn from an isotropic multivariate Gaussian with mean θ and a standard deviation σ, ϵ_i is additive Gaussian noise $\epsilon_i \sim \mathcal{N}(0, I)$ to a given parameter vector θ. Once the optimization step of ES is calculated, the return is added to the current parameter vector to obtain the next parameter vector.

The fundamental algorithm of POET is illustrated in Algorithm 1. The idea is to maintain a list of active environment-agent pairs EA_List that begins with a single starting pair ($E_{init}(\cdot)$, θ_{init}), where E_{init} is a simple recommendation environment and θ_{init} is a randomly-initialized weight vector (e.g. for a neural network). The POET then performs three main tasks at each iteration of its main loop: (1) generating new recommendation environments E($\cdot$) from those currently active through mutation of environment parameters. To generate a new environment, POET randomly perturbs the parameter vector of an active environment such that paired agents in the originating (parent) environments have exhibited sufficient progress to suggest that reproducing their respective environments would not be a waste of effort. The novel environments

Algorithm 1: Training Algorithm

```
Input: $E_{init}$: Initial Recommendation Environment
$\theta_{init}$: Initial Paired Agent denoted by policy parameter vector
$\alpha$: Learning Rate
$\sigma$: Noise Standard Deviation
T: Iterations
$T_{mutate}$: Time steps between Mutation
$T_{transfer}$: Time steps between Transfer
Output: EA_List
1  Initialize: EA_List = $\emptyset$
2  $\theta_{init}$ with random weights
3  Add ($E_{init}(.)$, $\theta_{init}$) to EA_List
4  for iteration t = 0...(T-1) do
5      if t > 0 and t mod $T_{mutate}$ = 0 then
6          Create new recommendation environments by mutation and append to EA_List
7      end
8      M = len(EA_List)
9      for m = 1...(M) do
10         Optimise each agent with its paired environment using ES
11     end
12     for m = 1...(M) do
13         if M > 1 and t mod $T_{transfer}$ = 0 then
14             Evaluate $\theta_1, ..., \theta_M$ on target environment E(.)
15             Return the top performing agent $\theta = \theta_{top}$
16             Perform one ES_step with all other
17         end
18     end
19 end
```

that are generated are not added to the current population of environments unless they are neither too hard nor too easy for the current population, (2) optimizing paired agents within their respective environments ES_step. The fact that each agent-environment pair is being optimized independently affords easy parallelization, wherein all the optimization steps can in principle be executed at the same time. and (3) attempting to transfer current agents θ from one environment to another. Given a list of input candidate agents $\theta_1 ... \theta_M$ and a target environment E($\cdot$), POET carries out two types of transfer attempts a) direct transfer, wherein agents from the originating environment are directly evaluated in the target environment, and b) proposal transfer, where agents take one ES optimization step in the target environment and the one with the highest reward is returned as θ_{top}. It is always possible that progress in one environment could end up helping in another. If the paired agent θ^A in environment $E^A(\cdot)$ is stuck in a local optimum, one remedy could be a transfer from the paired agent θ^B in environment $E^B(\cdot)$.

Fig. 1 illustrates the architecture of the policy network that is encoded by a feedforward DNN, where the input is a state s_t and the output has N^t task specific branches. Each branch has N^i task-specific layers and captures the distinct part of $\pi_S^{(i)}(\cdot)$. All the

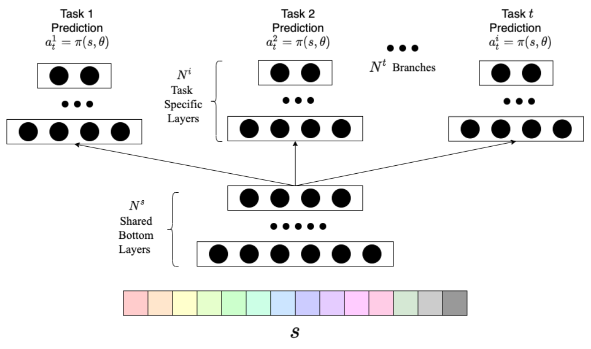

Figure 1: DNN Architecture of Policy Network

branches share the first N^s layers at the bottom to facilitate knowledge sharing among different recommendation tasks. The output of the i-th branch is a_t^i where the actions are determined by a stochastic policy function $a_t^i = \pi(s; \theta)$.

5 EXPERIMENTS

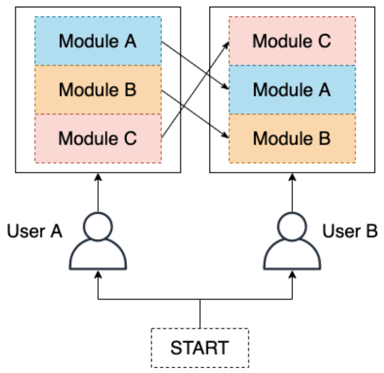

Figure 2: Personalization on ZOZOTOWN using module ranking

Simultaneously operating multiple recommendation tasks with long-term rewards is an important business requirement for many platforms to increase revenue and improve user experience. Unfortunately, to the best of our knowledge, all publicly available recommendation-related datasets are not suitable for evaluating the performance of our open-ended multi-task recommender. This is mainly because these datasets lack multiple-type feedbacks of recommendations, and the sequential dependency of those feedbacks that is captured in our model. Therefore, in this paper, we utilize the data from ZOZOTOWN E-Commerce platform, one of Japan's largest Fashion E-commerce platform, to conduct comprehensive experiments for evaluating the performance of our method. Fig. 2 illustrates the interface of module recommendation. Each module represented in red box is a collection of items. The rank of the modules should be personalized according to the user's interests in order to drive engagement and revenue. The home-page of the ZOZOTOWN recommends modules to users and collects different types of sequentially-dependent feedbacks: impressions, click and buy. Our recommendation goal is to simultaneously operate three recommendation tasks that maximize long-term impressions,clicks and buys respectively. We would like to note that our method is applicable to other applications as long as there are multiple recommendation tasks and long-term rewards to consider.

5.1 Reinforcement Learning modelling

We make use of RecSim [19] to design multi-task recommendations. We use latent state environment that models the long term user behavior [32]. This environment depicts a situation in which a user of an online service interacts with items of content, which are characterized by their observable quality on a scale of 0 to 1. In particular, items on 0-end (choc) generate engagement, but lead to decrease in long-term satisfaction but items on 1-end (kale) increase satisfaction but do not generate as much engagement. The challenge is to balance the two in order to achieve long-term optimal trade-off. The dynamics of this system are partially observable, as satisfaction is a latent variable. It has to be inferred through the increase/decrease in engagement. It also important to improve business revenue, hence we borrow ideas proposed in [34] to map the actions of the user such as impressions etc. to monetary value and provide suitable recommendations. In the further part of this subsection we will discuss the design of the simulator with respect to the above environment.

Document Model: Samples modules from prior distribution of module features. In the latent state environment, the modules are represented by a latent feature, namely their clickbaitness score. We also use CVR(Conversion rate) scores and impressions as the module features. We use Beta Distribution to model each feature. The document sampler samples modules from these three distributions.

User Model: Samples users from prior distribution of user features. Each user has a feature called net positive exposure (npe_t), and satisfaction (sat_t). Since satisfaction cannot be unbounded, they are related by logistic function.

$$sat_t = \mathcal{L}(\tau \cdot npe_t) \tag{2}$$

where τ is a user-specific sensitivity parameter, and $\mathcal{L}$ is sigmoid function. Once the user chooses a document, the net positive exposure evolves as

$$npe_{t+1} = \beta \cdot npe_t + 2(C - 0.5) + \mathcal{N}(0, \eta) \tag{3}$$

where β is some user-specific memory discount (forgetting factor), C is clickbaitness score and η is innovation standard deviation. This is the User Transition Model. Finally, the user engages with the chosen content for s_d seconds given by

$$s_d \sim logN(C \cdot \mu_c + (1 - C)\mu_k * sat_t, C \cdot \sigma_c + (1 - C) * \sigma_k) \tag{4}$$

where s_d is drawn according to a log-normal distribution with parameters linearly interpolating between the pure kale response (μ_k, σ_k) and the pure choc response (μ_c, σ_c). Thus, a user state is hence defined by the tuple $(sat_t, \tau, \beta, \eta, \mu_k, \sigma_k, \mu_c, \sigma_c)$. The only dynamic part of the User's state is satisfaction. Remember that other static variables help us to simulate different users.

User Choice Model: To simulate user response on ZOZOTOWN, we develop a multinomial choice model based on a simple Catboost model that uses User State and Module features and predicts the probability of the user's action for each task.

Action: After each user request the agent or the ranking model outputs predictions for each task, then the weighted-multiplication based ranking module combines these predicted scores to a final score through a combination function as shown in the Equation 5 and recommends top ranked modules.

Reward Function: The reinforcement learning problem represents goals by cumulative rewards. In addition, rewards also provide intermediate feedback(positive or negative) on the progress towards the goal. In our setup, our goal is to improve the overall engagement of the users on ZOZOTOWN by providing multi-task recommendations. Overall reward function consists of two components:

(1) Ranking based reward: The agent predict coefficients of ranking formula α, β, γ as in [34]. We then use the predicted coefficients to calculate rank score for each module. The rankscore for module i is given by,

$$rankscore(i) = CTR(i)^{\alpha} \times CVR(i)^{\beta} \times impressions(i)^{\gamma} \quad (5)$$

We then extract top K largest rank scores and assign a discounted weight $W_{\pi(i)}$ for each position and higher position has greater weight where $W_{\pi(i)} = exp(\pi(i))$. Finally ranking reward is calculated as the weighted sum of module rewards.

$$R_{ranking} = \sum rankscore(i) * W_{\pi(i)} \quad (6)$$

(2) Engagement based reward: Once the choice model predicts user's response for agent's recommendation user engages with the chosen module for s_d seconds according to the Equation 4. The engagement time provides feedback on user's satisfaction towards agent's recommendation. Hence we use $R_{engagement} = s_d$ as a reward.

Ranking-based rewards are rewards for ranking modules, and engagement-based rewards are rewards for interacting with modules. The final reward is the sum of ranking-based rewards and engagement-based rewards.

5.2 Training Procedure

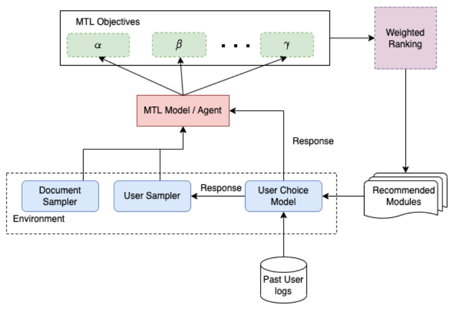

Figure 3: A detailed illustration of MTL ranking system

The entire learning process is parallelized using the Fiber Framework [59]. Generating new environments is how POET continues to produce new challenges. Each paired agent represented by Neural Network learns a policy which maps state (User's state and Module's state simulated by RecSim) to action (coefficients of ranking formula) that maximizes the expected reward. We restrict EA_List to store 20 active environments and their pair agents. The ANNECS score [50] is a counter for the number of environments in which the new environment has cleared the Minimal Criterion Coevolution(MCC) [5] process and has exceeded the reward threshold is used to measure the progress of learning. As the training progresses the ANNECS score increases as illustrated in Fig. 4 indicating that the algorithm is increasingly creating meaningful challenges. We then evaluate each of these Neural Networks against held out offline data for the Learning to Rank task and choose the best performing Neural Network as our model. Finally, we compare it with several other learning to rank models and observe that recommender systems trained using open-ended learning achieve better results than other methods.

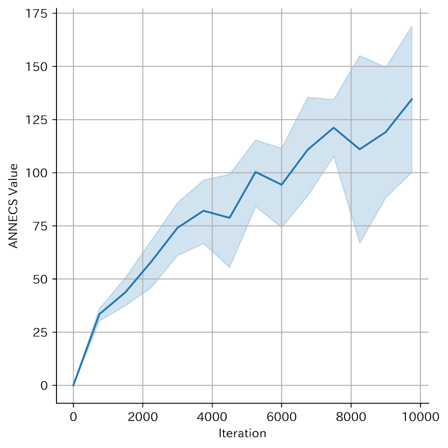

Figure 4: A plot of the ANNECS metric across iterations of POET recommender agent. Solid lines denotes the mean across 5 runs and shading denotes the 95% bootstrapped confidence intervals. The increasing curve indicates the new environments that can pass the Minimal Criterion Coevolution(MCC) continue to be consistently discovered.

5.3 Evaluation

Dataset Details To the best of our knowledge, there is no publicly available E-Commerce dataset that contains important features such as price and the labels of impression, click and purchase at the same time. Therefore, we collect a real-world dataset of E-Commerce recommendation from a popular E-Commerce platform. Due to the huge amount of online data, we collect two-week data, split the interaction data along time and use 80% of the data as training dataset for supervised methods and the rest 20% with samples over twenty-four million impressions for offline evaluations of all the methods. The dataset will be released to the public to support future studies. The basic statistics of the data is shown in the Table. 1. The dataset contains the log of recommended items, user's multiple types of feedbacks (i.e., impression, click, and buy) and the timestamps of receiving the feedback.

Stage	Users	Modules	Items	Impressions	Clicks	Buys
Training	3,996,837	53	66,439,403	67,167,616	2,508,352	928,535
Offline Evaluation	929,497	53	16,364,385	24,876,895	783,860	241,178

Table 1: Dataset Statistics

Baselines The main goal of this work is to demonstrate how open-ended learning can be used to simulate real-world multi-task recommendation. Once the online training using the simulator is completed we are left with 20 Environments and the parameters of Neural Networks of their paired agents. We then evaluate each of these Neural Networks against held out offline data for Learning to Rank with three other tasks and choose the best performing Neural Network as our model. Finally, we compare it with several other learning to rank and multi-task recommendation methods and observe that recommender systems trained using open-ended learning achieve better results than other direct methods.

We compare the our method with other baseline methods that can be grouped into three different categories: (1) traditional recommendation methods that are based on single-task supervised learning, (2) deep reinforcement learning methods that are based on single-task reinforcement learning, and (3) MTL methods that are based on multi-task supervised learning. The effectiveness of all competing methods is evaluated using Precision@K, NDCG@K, and MAP@K, which are the standard metrics used in previous research on recommender systems.

First category of algorithms include logistic regression(LR) and factorization machines(FM). The LR method estimates the probability of user liking an item through logistic regression over user and item features. In LR, only first order dependency between features and output are investigated. FMs [35] and their extensions provide a popular solution to using second-order feature interaction. We also evaluate with BPR-MF [37] which is one of the most widely used ranking methods for top-K recommendation and models recommendation as a pair-wise ranking problem. Due to their ability to incorporate high-order feature interactions, deep learning techniques have become the preferred method for working on recommendation system tasks. DNNs usually involve implicit nonlinear transformations of input features through a hierarchical structure of neural networks. We use Neural Collaborative filtering(NCF) [17], fusing both Generalized Matrix Factorization (GMF) and Multiple Layer Perceptron(MLP) under the NCF framework. Second category includes Evolutionary Strategies (ES) as proposed in Salimans et al.[38]. Third category includes MTL based methods like Feature-selected MTL(FMTL) and Mixture of Experts(MMOE).The DNNs for FMTL is arranged such that the first 2 layers are shared, and then the shared layers are connected to three task-specific branches, and outputs a predicted value for one task. The input to the DNNs is the concatenated features of a user and an item. The output of one branch is the estimated probability that the user likes the item in some task. The loss function of FMTL methods is composed of the cross-entropy loss plus an additional regularization term to encourage knowledge sharing among tasks. We use a multi-gate variant of Mixture of Experts(MMOE) [28] in which gating networks take the input features and output softmax gates assembling the experts with different weights, allowing different tasks to utilize experts differently. MMoE explicitly models the task relationships and learns task-specific functionalities to leverage shared representations. Note that the methods in the third category only need to be trained once to handle three tasks. They take into account the potential of knowledge-sharing in improving the performance over individual recommendation tasks. Compared to our method, the methods in the third category fail to consider the long-term rewards in ranking items to be recommended. These baseline methods provide a great representation for the state of the art.

Evaluation Metrics We adopt Normalised Discounted Cumulative Gain(NDCG@K) and Mean Average Precision(MAP@K) as metrics to evaluate ranking, which are widely used in recommendation systems. By default we set K=10 and report mean values of the above metrics for all the users during the testing phase.

5.4 Results Discussion

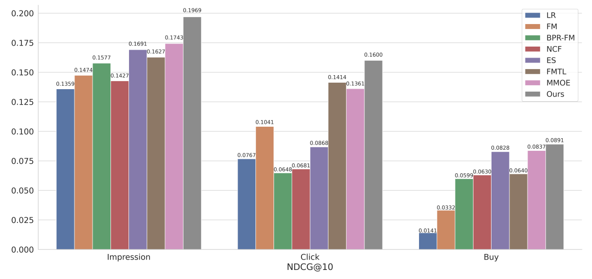

Figure 5: Bar plot comparing NDCG@K with K=10 of different recommendation methods for three tasks (Impression, Click, Buy). Our open-ended trained multi-task recommender(Ours) outperforms other supervised and reinforcement learning based methods in all the three tasks.

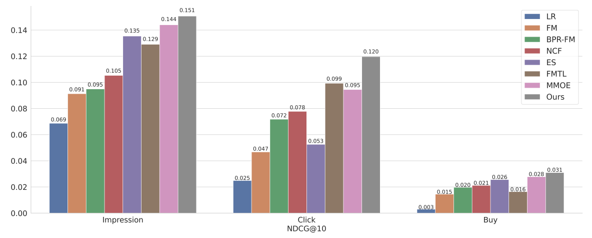

Figure 6: Bar plot comparing MAP@K with K=10 of different recommendation methods for three tasks (Impression, Click, Buy). Our open-ended trained multi-task recommender(Ours) outperforms other supervised and reinforcement learning based methods in all the three tasks.

To evaluate the effectiveness of our open-ended learning based recommender, we compare it with seven competitors over two standard evaluation metrics (NDCG@K and MAP@K). Fig. 5 and Fig. 6 show the performance of against LR, FMs and NCF when K = 10 on three types of feedback impression, click, and buy, respectively. It is interesting to note that, simpler methods such as FM perform better than NCF over the impression optimization task and click optimization task but underperform NCF in the buy optimization task. This happens likely due to the fact that the difficulty decreases from the click optimization task to the impression optimization task to the buy optimization task as the data imbalance and label sparsity are worse from the latter to the former.

As a comparison, LR and FMs assume the output linearly depends on the first-order or second-order interactions of the features, the underlying assumptions of which are more likely to hold in simpler tasks. Another observation is that our method substantially outperforms all the three baseline methods over all the recommendation tasks and all evaluation metrics. Its performance is also relatively stable among tasks of different levels of difficulties compared to other competitors. The reason is that all three baseline methods are based on supervised learning and thus unable to plan the recommendations in a way that considers the long-term reward. They also follow the single-task learning framework and thus fail to take advantage of the knowledge from other tasks to improve its performance of each task. On the contrary, open-ended learning agent makes recommendations by optimizing the rewards in the longer horizon and encourages knowledge-sharing through jointly learning multiple recommendation tasks.

Fig. 5 illustrates the performance of open-ended learning based method(Ours) against ES over the same collection of recommendation tasks and evaluation metrics. The ES method is based on single-task deep reinforcement learning. The open-ended learning recommender agent shows better performance in all recommendation tasks, this may be due to the no fact that the stepping stones that lead to solutions to very challenging environments are more likely to be found through a divergent, open-ended process than through a direct attempt to optimize in the challenging environment. Another interesting observation is that compared to the supervised learning baselines presented earlier the ES method demonstrates more stable performance over different tasks with the same evaluation metrics. The reason is as aforementioned: compared to SL-based recommenders that optimize short-term rewards, the RL-based recommenders plan their recommendations to optimize a long-term goal that may overcome the uncertainty when task difficulty varies.

The regularization term in FMTL allows for different tasks to learn task-specific features. In other words, FMTL offers more freedom in knowledge transfer from easy tasks to difficult tasks as compared to other methods discussed earlier. MMoE's gating networks can effectively modularize input information into experts for task relation and confict modeling. Compared to FMTL, MMOE seems to favor more difficult tasks than easy ones in knowledge transfer. As illustrated, One crucial observation is that these two baselines outperform LR, FMs and NCF, baselines based on single-task supervised learning, over almost all tasks and evaluation metrics. This advantage is largely because of the knowledge-sharing between different tasks. It is also worth emphasizing that our method achieves outstanding performance over all tasks and almost all evaluation metrics compared to the two MTL baselines. The reason is that although both open-ended learning and the baselines follow MTL framework, our method aims to maximize long-term rewards in making recommendations and takes into account the impacts of current recommendation to future rewards.

6 CONCLUSION

In this paper, we started with the description of a few real-world challenges in designing and developing industrial recommendation systems, especially ranking systems. These challenges include the presence of multiple competing ranking objectives, optimising for long term rewards and simulator to real world adaptation. To tackle these challenges, we proposed a multi-task ranking system that uses open-ended learning and applied it to the problem of recommending modules. The ability of the proposed method to automatically build curriculum of increasingly complex recommendation environments and simultaneously optimize on the ranking objectives makes it viable for Simulator to Real world transfer. We evaluated our method using a large-scale dataset collected from experiments over Japan's Largest Fashion E-commerce platform ZOZOTOWN. The evaluation results using multiple metrics demonstrate better effectiveness and efficiency of our developed method against several state-of-the-art methods.

REFERENCES

[1] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528* (2019).

[2] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. 2011. Matrix Factorization Techniques for Context Aware Recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems* (Chicago, Illinois, USA) *(RecSys '11)*. Association for Computing Machinery, New York, NY, USA, 301–304. https://doi.org/10.1145/2043932.2043988

[3] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *proceedings of the 10th ACM Conference on Recommender Systems*. 107–114.

[4] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowl. Based Syst.* 46 (2013), 109–132.

[5] Jonathan C Brant and Kenneth O Stanley. 2017. Minimal criterion coevolution: a new approach to open-ended search. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 67–74.

[6] Biao Chang, Hengshu Zhu, Yong Ge, Enhong Chen, Hui Xiong, and Chang Tan. 2014. Predicting the Popularity of Online Serials with Autoregressive Models. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (Shanghai, China) *(CIKM '14)*. Association for Computing Machinery, New York, NY, USA, 1339–1348. https://doi.org/10.1145/2661829.2662055

[7] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2019. Large-scale Interactive Recommendation with Tree-structured Policy Gradient. In *AAAI*.

[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 *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM '19)*. Association for Computing Machinery, New York, NY, USA, 456–464. https://doi.org/10.1145/3289600.3290999

[9] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing Reinforcement Learning in Dynamic Environment with Application to Online Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1187–1196. https://doi.org/10.1145/3219819.3220122

[10] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten De Rijke. 2019. Joint Neural Collaborative Filtering for Recommender Systems. *ACM Trans. Inf. Syst.* 37, 4, Article 39 (aug 2019), 30 pages. https://doi.org/10.1145/3343117

[11] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah.

2016. Wide Deep Learning for Recommender Systems. *arXiv:1606.07792* (2016). http://arxiv.org/abs/1606.07792

[12] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. 2015. Robots that can adapt like animals. *Nature* 521, 7553 (2015), 503–507.

[13] Dario Floreano and Claudio Mattiussi. 2008. Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies (Intelligent Robotics and Autonomous Agents). (2008).

[14] Izzeddin Gur, Natasha Jaques, Yingjie Miao, Jongwook Choi, Manoj Tiwari, Honglak Lee, and Aleksandra Faust. 2021. Environment Generation for Zero-Shot Compositional Reinforcement Learning. *Advances in Neural Information Processing Systems* 34 (2021).

[15] Nicholas Guttenberg, Nathaniel Virgo, and Alexandra Penn. 2019. On the potential for open-endedness in neural networks. *Artificial life* 25, 2 (2019), 145–167.

[16] Guy Hadash, Oren Sar Shalom, and Rita Osadchy. 2018. Rank and rate: multi-task learning for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems.* 451–454.

[17] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval.* 355–364.

[18] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569

[19] Eugene Ie, Chih-wei Hsu, Martin Mladenov, Vihan Jain, Sanmit Narvekar, Jing Wang, Rui Wu, and Craig Boutilier. 2019. Recsim: A configurable simulation platform for recommender systems. *arXiv preprint arXiv:1909.04847* (2019).

[20] 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 *Proceedings of the 28th International Joint Conference on Artificial Intelligence* (Macao, China) *(IJCAI'19)*. AAAI Press, 2592–2599.

[21] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation* 3, 1 (1991), 79–87.

[22] How Jing and Alexander J Smola. 2017. Neural survival recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining.* 515–524.

[23] Joel Lehman and Kenneth O Stanley. 2011. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation* 19, 2 (2011), 189–223.

[24] Joel Lehman and Kenneth O Stanley. 2012. Beyond open-endedness: Quantifying impressiveness. In *ALIFE 2012: The Thirteenth International Conference on the Synthesis and Simulation of Living Systems.* MIT Press, 75–82.

[25] Joel Lehman, Kenneth O Stanley, et al. 2008. Exploiting open-endedness to solve problems through the search for novelty.. In *ALIFE*. Citeseer, 329–336.

[26] Yu Lei and Wenjie Li. 2019. Interactive Recommendation with User-Specific Deep Reinforcement Learning. *ACM Trans. Knowl. Discov. Data* 13, 6, Article 61 (oct 2019), 15 pages. https://doi.org/10.1145/3359554

[27] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like it: multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems.* 4–12.

[28] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 1930–1939.

[29] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* 1137–1140.

[30] Nicola Milano and Stefano Nolfi. 2021. Automated curriculum learning for embodied agents a neuroevolutionary approach. *Scientific Reports* 11, 1 (2021), 1–14.

[31] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 3994–4003.

[32] Martin Mladenov, Ofer Meshi, Jayden Ooi, Dale Schuurmans, and Craig Boutilier. 2019. Advantage Amplification in Slowly Evolving Latent-State Environments. *CoRR* abs/1905.13559 (2019). http://arxiv.org/abs/1905.13559

[33] Harrie Oosterhuis and Maarten de Rijke. 2018. Ranking for relevance and display preferences in complex presentation layouts. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval.* 845–854.

[34] Changhua Pei, Xinru Yang, Qing Cui, Xiao Lin, Fei Sun, Peng Jiang, Wenwu Ou, and Yongfeng Zhang. 2019. Value-aware recommendation based on reinforced profit maximization in e-commerce systems. *arXiv preprint arXiv:1902.00851* (2019).

[35] Steffen Rendle. 2010. Factorization Machines. *2010 IEEE International Conference on Data Mining* (2010), 995–1000.

[36] Steffen Rendle. 2012. Factorization Machines with LibFM. *ACM Trans. Intell. Syst. Technol.* 3, 3, Article 57 (may 2012), 22 pages. https://doi.org/10.1145/2168752.2168771

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618* (2012).

[38] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864* (2017).

[39] Russell K Standish. 2003. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications* 3, 02 (2003), 167–175.

[40] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[41] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM Conference on Recommender Systems.* 269–278.

[42] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 565–573. https://doi.org/10.1145/3159652.3159656

[43] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, et al. 2021. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808* (2021).

[44] Georgios Theocharous, Philip S Thomas, and Mohammad Ghavamzadeh. 2015. Personalized ad recommendation systems for life-time value optimization with guarantees. In *Twenty-Fourth International Joint Conference on Artificial Intelligence.*

[45] Katrien Verbert, Nikos Manouselis, Xavier Ochoa, Martin Wolpers, Hendrik Drachsler, Ivana Bosnic, and Erik Duval. 2012. Context-Aware Recommender Systems for Learning: A Survey and Future Challenges. *IEEE Transactions on Learning Technologies* 5, 4 (2012), 318–335. https://doi.org/10.1109/TLT.2012.11

[46] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) *(KDD '15)*. Association for Computing Machinery, New York, NY, USA, 1235–1244. https://doi.org/10.1145/2783258.2783273

[47] Jialei Wang, Steven CH Hoi, Peilin Zhao, and Zhi-Yong Liu. 2013. Online multi-task collaborative filtering for on-the-fly recommender systems. In *Proceedings of the 7th ACM conference on Recommender systems.* 237–244.

[48] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha. 2018. Supervised Reinforcement Learning with Recurrent Neural Network for Dynamic Treatment Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 2447–2456. https://doi.org/10.1145/3219819.3219961

[49] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O Stanley. 2019. Paired open-ended trailblazer (poet): Endlessly generating increasingly complex and diverse learning environments and their solutions. *arXiv preprint arXiv:1901.01753* (2019).

[50] Rui Wang, Joel Lehman, Aditya Rawal, Jiale Zhi, Yulun Li, Jeffrey Clune, and Kenneth Stanley. 2020. Enhanced POET: Open-ended reinforcement learning through unbounded invention of learning challenges and their solutions. In *International Conference on Machine Learning.* PMLR, 9940–9951.

[51] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-Hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 55–64. https://doi.org/10.1145/3077136.3080809

[52] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep Item-Based Collaborative Filtering for Top-N Recommendation. *ACM Trans. Inf. Syst.* 37, 3, Article 33 (apr 2019), 25 pages. https://doi.org/10.1145/3314578

[53] Yu Zhang and Qiang Yang. 2017. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114* (2017).

[54] Hongke Zhao, Qi Liu, Guifeng Wang, Yong Ge, and Enhong Chen. 2016. Portfolio Selections in P2P Lending: A Multi-Objective Perspective. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 2075–2084. https://doi.org/10.1145/2939672.2939861

[55] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems.* 95–103.

[56] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 1040–1048. https://doi.org/10.1145/3219819.3219886

[57] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. 2019. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*. 43–51.

[58] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference*. 167–176.

[59] Jiale Zhi, Rui Wang, Jeff Clune, and Kenneth O Stanley. 2020. Fiber: A platform for efficient development and distributed training for reinforcement learning and population-based methods. *arXiv preprint arXiv:2003.11164* (2020).