Reinforcing User Retention in a Billion Scale Short Video Recommender System

Qingpeng Cai
Kuaishou Technology
Beijing, China
caiqingpeng@kuaishou.com

Shuchang Liu*
Kuaishou Technology
Beijing, China
liushuchang@kuaishou.com

Xueliang Wang
Kuaishou Technology
Beijing, China
wangxueliang03@kuaishou.com

Tianyou Zuo
Kuaishou Technology
Beijing, China
zuotianyou@kuaishou.com

Wentao Xie
Kuaishou Technology
Beijing, China
xiewentao@kuaishou.com

Bin Yang
Kuaishou Technology
Beijing, China
yangbin11@kuaishou.com

Dong Zheng
Kuaishou Technology
Beijing, China
zhengdong@kuaishou.com

Peng Jiang†
Kuaishou Technology
Beijing, China
jiangpeng@kuaishou.com

Kun Gai
Unaffiliated
Beijing, China
gai.kun@qq.com

ABSTRACT

Recently, short video platforms have achieved rapid user growth by recommending interesting content to users. The objective of the recommendation is to optimize user retention, thereby driving the growth of DAU (Daily Active Users). Retention is a long-term feedback after multiple interactions of users and the system, and it is hard to decompose retention reward to each item or a list of items. Thus traditional point-wise and list-wise models are not able to optimize retention. In this paper, we choose reinforcement learning methods to optimize the retention as they are designed to maximize the long-term performance. We formulate the problem as an infinite-horizon request-based Markov Decision Process, and our objective is to minimize the accumulated time interval of multiple sessions, which is equal to improving the app open frequency and user retention. However, current reinforcement learning algorithms can not be directly applied in this setting due to uncertainty, bias, and long delay time incurred by the properties of user retention. We propose a novel method, dubbed RLUR, to address the aforementioned challenges. Both offline and live experiments show that RLUR can significantly improve user retention. RLUR has been fully launched in Kuaishou app for a long time, and achieves consistent performance improvement on user retention and DAU.

*The first two authors contributed equally to this work
†Corresponding author

WWW'23, April 30 - May 4, 2023, Austin, Texas, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

CCS CONCEPTS

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

KEYWORDS

short video, user retention, reinforcement learning

ACM Reference Format:
Qingpeng Cai, Shuchang Liu, Xueliang Wang, Tianyou Zuo, Wentao Xie, Bin Yang, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Reinforcing User Retention in a Billion Scale Short Video Recommender System. In *Proceedings of Proceedings of the ACM Web Conference 2023 (WWW'23).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

As newly emerging media and sharing platforms, short video applications like TikTok, YouTube Shorts, and Kuaishou quickly attract billions of users by recommending interesting content to them. There has been an increasing interest in short video recommendation [11, 15, 24, 29] in academia and industry. In the view of the recommender, a user interacts with the system through explicit and implicit feedback (e.g. watching time, like, follow, comment, etc.) in multiple requests, and the system returns a list of short videos at each request. The ultimate goal is to improve retention, which reflects user satisfaction [27, 33]. User retention is defined as the ratio of visiting the system again, and commonly referred to the retention at next day. It directly affects DAU, which is the core value of the short video app.

Current recommender systems deploy point-wise models [5] or list-wise models [20] to recommend items to users. Point-wise models predict the immediate reward of user-item pair, while list-wise models estimate the combination of rewards of a list of items considering the positions and relationships between items. Both of them aim at predicting a point-wise reward or a combination of point-wise rewards. However, retention reward is a long-term feedback after multiple interactions between users and the system. Similar to Go game [22], the relationship between retention reward and the intermediate feedback is not clear, and it is difficult to decompose retention reward to each item or a list of items. Thus traditional models are not able to optimize retention.

Reinforcement learning (RL) [23] methods are designed to learn a policy to maximize the long-term reward by interacting with the environment. Thus in this paper we choose RL methods to optimize retention. We model the problem of optimizing user retention in short video systems as an infinite horizon request-based Markov Decision Process (MDP), where the recommender is the agent, and users serve as the environments. The session starts when the user visits the app. At each step (the user's request), the agent plays an action (the ranking weight) to ensemble scores from scoring models that predict various user feedback. The ensemble ranking function inputs both the ranking weight and the prediction scores, outputs videos with highest scores to the user. Then the user provides immediate feedback including watch time and other interactions. The session ends when the user leaves the app, the next session starts when the user opens the app again and the process repeats. Our objective is to minimize the cumulative returning time (defined as the time gap between the last request of the session and the first request of the next session), which is equal to improving the frequency of user visits, i.e. app open frequency. And the frequency of user visits directly corresponds to retention. Different from previous RL for recommender system works [1, 3, 4, 8, 10, 16–18, 25, 28, 30–33] that maximizes the cumulative immediate feedback, we are one of the first works to directly optimize user retention in short video recommender systems to the best of our knowledge.

However, current RL algorithms can not be trivially applied due to the following properties of retention reward: 1) **Uncertainty**: retention is not totally decided by the recommendation algorithm, and is usually affected by many uncertain factors outside the system. For example, retention is disturbed by the noise of social events. 2) **Bias**: retention is biased with the different factors including time and the level of user's activity. The retention varies between weekdays and weekends, and highly active users inherently have high retention. 3) **Long delay time**: different from the instant reward signal in games, the retention reward returns mostly in several hours. It causes the instability of the training of RL policy due to huge distribution shifts [13].

For the uncertainty problem, we propose a novel normalization technique that predicts the returning time and use it to reduce the variance of the retention reward. For the bias problem, we train different policies over user groups to prevent the learning from being dominated by high active users. For the long delay time problem, we propose a novel soft regularization method to better balance the sample efficiency and stability. We also take the immediate feedback as the heuristic rewards and the intrinsic motivation methods [2, 19] to better optimize user retention in the delayed reward setting. Combining the above techniques, we propose the Reinforcement Learning for User Retention algorithm, named as RLUR.

We summarize our contributions as below:

- We model the user retention for short video recommendation problem as an infinite horizon requested-based MDP, and the aim is to minimize the cumulative returning time.
- We propose novel methods to solve the challenges of uncertainty, bias, and long delay time of user retention.
- Experiments on both offline and live environments show that RLUR improves user retention significantly.
- The RLUR algorithm has been fully launched in Kuaishou app, and it shows that RLUR continuously improves user retention and DAU.

2 PROBLEM DEFINITION

We model the problem as an infinite horizon request-based Markov Decision Process (MDP), where the recommender is the agent, and users are the environments. As shown in Figure 1(a), when the user opens the app, a session i starts. At each request (step) i_t of the session i, the agent plays an action a_{i_t} given the state s_{i_t} of the user. n deep models predict scores $x_j = (x_{j1}, ..., x_{jn})$ for each candidate video j, in terms of various feedback(watch time, follow, like, etc). The ranking function f inputs the action a_{i_t} and the prediction scores x_j, and outputs the ranking score $f(a_{i_t}, x_j)$ for each video j. The system recommends top 6 videos with the highest ranking score to the user as shown in Figure 1(b). Then the user provides immediate feedback $I(s_{i_t}, a_{i_t})$. The session ends when the user leaves the app. When the user returns to the app again, the next session $i+1$ starts, and the delayed reward(returning time) of session i is returned. Then the above process repeats.

We now introduce the detail of the MDP. The **state** s_{it} consists of user profile, behavior history u_{i_t}, request context, and candidate video features. The **action** is a continuous vector in $[0, C]^n$, where n is the number of scoring models. We use a linear **ranking function**, i.e, $f(a_{i_t}, x_j) = \sum_{k=1}^{n} a_{i_t k} x_{jk}$. The **immediate reward** $I(s_{i_t}, a_{i_t})$ is defined as the sum of watch time and the number of interactions at this request. The **returning time** $T(s_i)$ is the time gap between the last request of session s_i and the first request of session s_{i+1}. The **returning time reward** $r(s_{i_t}, a_{i_t})$ is $T(s_i)$ for the last request, and 0 otherwise. We learn a deterministic policy $\pi(s_{i_t}|\theta)$ that inputs the state s_{i_t} and outputs the action a_{i_t}. The objective is to minimize the cumulative returning time $\sum_{i=1}^{\infty} \gamma^{i-1} T(s_i)$, where $\gamma (0 < \gamma < 1)$ is the discount factor.

3 METHOD

In this section we firstly discuss the learning of retention in the delayed reward setting. Then we propose techniques to tackle these challenges incurred by the characteristic of user retention.

3.1 Retention Critic Learning

In this section we discuss how to learn the retention. We learn a critic function $Q_T(s_{i_t}, a_{i_t}|w_T)$ parameterized by w_T, to estimate the cumulative returning time from the current state s_{i_t} and action a_{i_t}. We follow the deep deterministic policy gradient method [14] to estimate the cumulative returning time reward. As shown in Figure 1(d), the loss function $L(w_T)$ for learning returning time is

$$\sum_{s_{i_t}, a_{i_t} \in D} (Q_T(s_{i_t}, a_{i_t}|w_T) - (r(s_{i_t}, a_{i_t}) + \gamma_{i_t} Q_T(s_{i_{t+1}}, \pi(s_{i_{t+1}}|\theta)|w_T)))^2, \tag{1}$$

where the discount factor γ_{i_t} is 1 for samples that are not the last requests and γ for the last requests, and D is the dataset. The discount

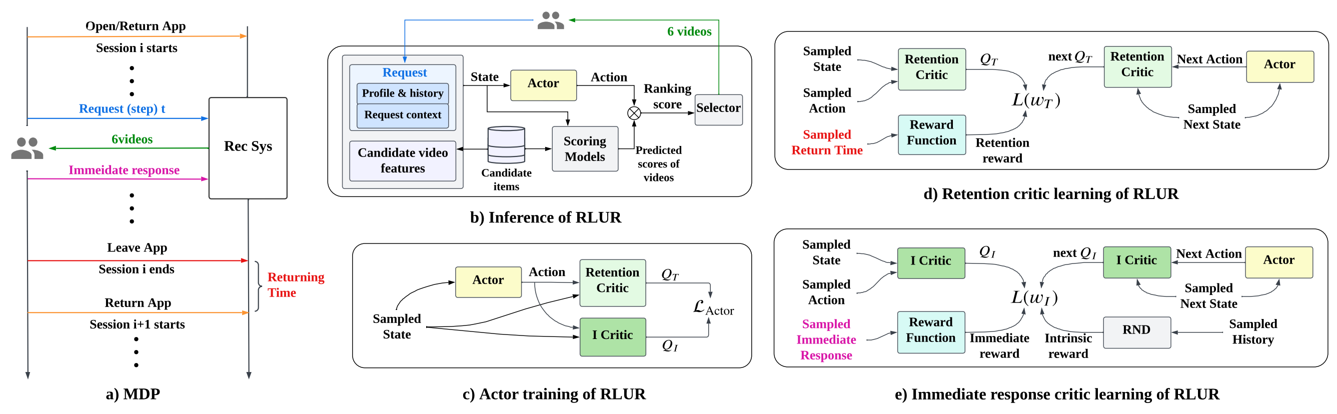

Figure 1: The infinite horizon request-based MDP and the framework of RLUR

factor of non-terminal samples is set as 1 to prevent the returning time reward from vanishing by the exponential decay mechanism. If one set the discount factor to be a number less than 1, the importance of returning time in future sessions will be extremely small. Optimizing the loss (1) is equivalent to estimate $\sum_{i=1}^{\infty}\gamma^{i-1}T(s_i)$. We omit the proof due to lack of space.

3.2 Methods for Delayed Reward

The returning time reward only occurs at the end of each session, and it is inefficient to learn an RL policy with delayed rewards. For better learning of user retention, we adopt the heuristic reward methods [12] to enhance the policy learning and the intrinsic motivation methods [2, 19] to drive the policy to explore novel states. As the feedback and returning time are positively correlated, we take the immediate rewards as heuristic rewards to guide the policy to improve the user retention. For better exploration, we choose the Random Network Distillation (RND) [2] method that is both effective and computationally efficient. The idea is that we randomly initialize two networks with the same structure, and train one network to fit the output of another fixed network. The loss of RND is $L(w_e) = \sum_{u_{i_t}\in D}||E(u_{i_t}|w_e) - E(u_{i_t}|w_e^*)||_2^2$, where E is the embedding function of behavior history u_{i_t}, w_e is the parameter, and w_e^* is the fixed parameter. Intuitively, the loss of each state decreases with more samples. Thus it can quantify the novelty of each state. We use the RND loss of each sample as the intrinsic reward. To reduce the interference of immediate rewards with the retention reward, we learn a separate critic function, the immediate reward critic $Q_I(s_{i_t}, a_{i_t}|w_I)$ to estimate the sum of the intrinsic rewards with the immediate rewards, $r_I(s_{i_t}, a_{i_t}) = I(s_{i_t}, a_{i_t}) + ||E(u_{i_t}|w_e) - E(u_{i_t}|w_e^*)||_2^2$. The loss of the immediate reward critic, $L(w_I)$ is similar to Eq.(1), shown in Figure 1(e).

3.3 Uncertainty

The returning time is highly uncertain and affected by many factors outside the system. To reduce its variance , we now propose a normalization technique. We use the ratio of the true returning time over the predicted returning time as the normalized retention reward. For predicting the returning time, we learn a session-level classification model T' to predict the returning probability. We firstly calculate the empirical distribution of returning time, then use the $\beta\%$ percentile of the distribution, T_β to determine whether the sample x is positive or negative. The label y of sample x is positive if the returning time is less than T_β as the shorter time means the better. The loss function of T' is $y * logT'(x) + (1 - y) * log(1 - T'(x))$, and $T'(x)$ predicts the probability that the returning time is shorter than T_β. Then we get a lower bound of the expected returning time by the Markov Inequality [26], $ET(x) \geq P(T(x) \geq T_\beta) * T_\beta \sim (1 - T'(x)) * T_\beta$ as the predicted returning time. Thus we get the normalized retention reward, $r(s_{i_t}, a_{i_t}) = clip\{0, \frac{T(s_i)}{(1-T'(x))*T_\beta}, \alpha\}$, where α is a positive constant.

3.4 Bias

As the returning time of high active users is much shorter than that of low active users, and the habit of these groups are quite different, we learn two separate policies $\pi(\cdot|\theta_{\text{high}})$ and $\pi(\cdot|\theta_{\text{low}})$ for high active group and low active group respectively. As we want to minimize the returning time and maximize the immediate rewards, the loss of the policy $\pi(\cdot|\theta_{\text{high}})$ is a weighted sum of the immediate critic and retention critic,

$$L(\theta_{\text{high}}) = \lambda_T Q_T(s_{i_t}, \pi(s_{i_t}|\theta_{\text{high}})|w_T) - \lambda_I Q_I(s_{i_t}, \pi(s_{i_t}|\theta_{\text{high}})|w_I), \quad (2)$$

where λ_T, λ_I are positive weights. The loss of the other policy is similar. The learning of Actor is shown in Figure 1(c).

3.5 Tackling the Unstable Training Problem

Different from that the reward returns instantly in games, the retention reward returns in several hours to days, which causes a much larger distribution shift between current policy and behavior policy and the instability of the training of RL algorithms in our setting. Regularization methods [6] that adds a behavior cloning loss are proposed to stabilize the training in the off-policy setting. However, we find that this method either fails to stabilize or limits the sample efficiency. To better balance the sample efficiency and the stability, we now propose a novel soft regularization method. The actor loss is defined as $\exp(\max\{\lambda * (log(p(a_{i_t}|s_{i_t})) - log(p_b(a_{i_t}|s_{i_t}))), 0\})L(\theta)$,

Table 1: Offline Results

Algorithm	Returning time↓	User retention↑
CEM	2.036	0.587
TD3	2.009	0.592
RLUR (naive, $\gamma = 0$)	2.001	0.596
RLUR (naive, $\gamma = 0.9$)	1.961	0.601
RLUR	**1.892**	**0.618**

where $\lambda > 0$ is the regularization coefficient, $p(a_{i_t}|s_{i_t})$ is the probability density of the Gaussian distribution of the current policy, and $p_b(a_{i_t}|s_{i_t})$ is the probability density of the Gaussian distribution of the behavior policy. The intuition is that samples with higher distribution shift get less weight in the learning, which softly regularizes the learning. λ controls the regularization degree, larger λ makes a stronger regularization on the actor loss. Note that when λ is 0, we do not make any regularization on the actor loss.

4 OFFLINE EXPERIMENTS

We validate the effectiveness of RLUR on a public short video recommendation dataset, *KuaiRand* [9], which is an unbiased dataset that contains the logs of the interactions of users and the system with multiple sessions. We build a simulator based on this dataset, which includes three parts: the user immediate feedback module that predicts multiple user behaviors; the leave module that predicts whether the user leaves the session or not; and the return module that predicts the probability of returning to the app on day $k \in \{1, \ldots, K\}$ after each session. We set $K = 10$.

We compare RLUR with a black-box optimization method, the Cross Entropy Method (CEM) [21] that is commonly used for ranking parameter searching and a state of the art reinforcement learning method, TD3 [7]. We evaluate the performance of each algorithm in terms of the averaged returning day (returning time) and averaged retention on the 1st day (user retention) across all user sessions. We train each algorithm until convergence, and we report the averaged performance of the last 50 episodes in Table 1. For both metrics, TD3 outperforms CEM, which demonstrates the effectiveness of reinforcement learning. RLUR outperforms both TD3 and CEM significantly. We also train a variant of RLUR that only contains the part of learning the returning time in Section 3.1, called RLUR (naive). RLUR (naive, $\gamma = 0.9$) outperforms RLUR (naive, $\gamma = 0$), which shows that it is more reasonable to minimize the cumulative retention of multiple sessions than one session as γ controls the importance of future sessions. RLUR outperforms RLUR (naive, $\gamma = 0.9$) substantially, which validates the effectiveness of the proposed techniques for the retention challenge.

5 LIVE EXPERIMENTS

We test our proposed RLUR algorithm by live experiments in Kuaishou, a billion-scale short video platform. We compare RLUR with a black-box optimization baseline that is widely used for ranking parameter searching in recommender systems, CEM [21]. We do not compare TD3 here as the training of TD3 is unstable, which is discussed in Section 3.5. We randomly split users into two buckets, deploy RLUR in the test bucket, and run CEM in the base bucket. We evaluate the performance of algorithms in terms of app open frequency, DAU, user retention at 1st day, and user retention at 7th day. We test algorithms for a long time to get convincing results as DAU and user retention varies between days. Now we introduce the details.

Inference and Training. The inference procedure is shown in Figure 1(b). At each user request, the user state is sent to the Actor, and the mean μ and variance σ are returned. Then the action is sampled from a Gaussian distribution $N(\mu, \sigma)$. Then the ranking function calculates the linear product of the action with the predicted scores of each video, and recommends 6 videos with the highest scores to the user. The training of actor is illustrated in Figure 1(c). The actor is trained to minimize the weighted sum of the retention critic and the immediate reward critic, as in Eq.(2).

MDP. The state is a vector of user profile, behavior history(user statistics, the id of videos and corresponding feedback of the user in previous 3 requests), request context, and the candidate video features. The user profile covers age, gender, and location. The user statistics include statistics of various feedback. For the action, we choose an 8-dimensional continuous vector ranging in $[0, 4]^8$, and the policy outputs the parameters of eight scoring models predicting the main feedback (watchtime, shortview, longview, like, follow, forward, comment, and personal-page entering). The immediate reward $I(s_{it}, a_{it})$ of each request is designed as the the sum of watch time (in seconds) and interactions (including like, follow, comment, and share) of 6 videos.

Hyper-parameters. The discounted factor is 0.95, and 0.95 outperforms other values in our system. The weights in actor loss is set to be 1.0, 1.0. As for the session-level retention model, we choose 60% percentile to determine the label, and the value of upper bound, α is 3. The regularization coefficient λ is 1.5.

Figure 2 plot the comparison results of RLUR with CEM, where the x axis stands for the number of days after deployment, and the y axis shows the percentage performance gap between RLUR and CEM. As we can see, app open frequency which directly reflects the returning time, increases consistently from Day 0 to Day 80 and sharply from Day 80 to Day 100. The app open frequency converges to 0.450% after Day 100. That validates the training of RLUR can consistently increase the retention reward and converges at expected. The user retention at 1st day/7th day and DAU increase slowly with the training from Day 0 to Day 80. From day 80 to day 100, these metrics increases sharply. After day 100, the performance gap of DAU converges to 0.2%, user retention at 1st day converges to 0.053%, and user retention at 7th day converges to 0.063%. Note that 0.01% improvement of user retention and 0.1% improvement of DAU are statistically significant in short video platforms. That is, the performance of RLUR is quite significant. Note that DAU and retention continue to increase.

6 CONCLUSION

We study to optimize user retention in short video recommender systems. We discuss the challenges of optimizing user retention. We formalize the problem as an infinite-horizon request-based Markov Decision Process, and our aim is to learn a policy that minimizes the accumulated returning time. We propose novel techniques to tackle the challenge of retention, and we present the RLUR algorithm. Both offline and live experiments validate the effectiveness of RLUR. We

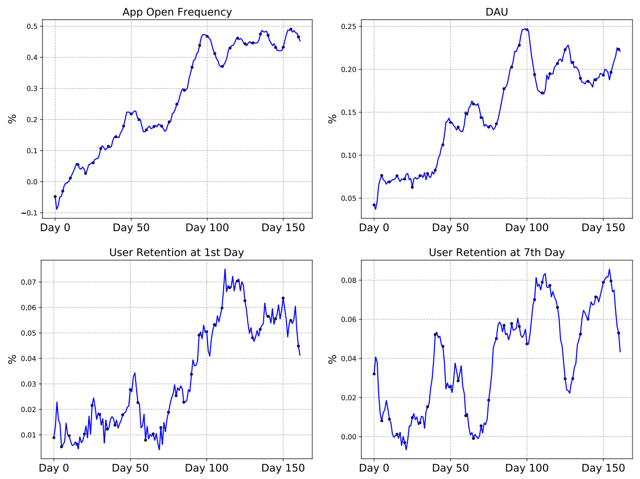

Figure 2: Live performance gap of each day.

have deployed RLUR in a billion-scale short video system for a long time, and it improves user retention and DAU significantly and consistently.

REFERENCES

[1] M Mehdi Afsar, Trafford Crump, and Behrouz Far. 2021. Reinforcement learning based recommender systems: A survey. *arXiv preprint arXiv:2101.06286* (2021).

[2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. 2018. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894* (2018).

[3] 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 *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3312–3320.

[4] 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 & Data Mining*. 1187–1196.

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[6] Scott Fujimoto and Shixiang Shane Gu. 2021. A minimalist approach to offline reinforcement learning. *Advances in neural information processing systems* 34 (2021), 20132–20145.

[7] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*. PMLR, 1587–1596.

[8] Chongming Gao, Wenqiang Lei, Jiawei Chen, Shiqi Wang, Xiangnan He, Shijun Li, Biao Li, Yuan Zhang, and Peng Jiang. 2022. CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System. *arXiv preprint arXiv:2204.01266* (2022).

[9] Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. 2022. KuaiRand: An Unbiased Sequential Recommendation Dataset with Randomly Exposed Videos. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*. 5 pages. https://doi.org/10.1145/3511808.3557624

[10] Yingqiang Ge, Shuchang Liu, Ruoyuan Gao, Yikun Xian, Yunqi Li, Xiangyu Zhao, Changhua Pei, Fei Sun, Junfeng Ge, Wenwu Ou, et al. 2021. Towards Long-term Fairness in Recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 445–453.

[11] Xudong Gong, Qinlin Feng, Yuan Zhang, Jiangling Qin, Weijie Ding, Biao Li, and Peng Jiang. 2022. Real-time Short Video Recommendation on Mobile Devices. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management* (Atlanta, GA, USA) *(CIKM '22)*.

[12] Marek Grzes. 2017. Reward shaping in episodic reinforcement learning. (2017).

[13] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643* (2020).

[14] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[15] Zihan Lin, Hui Wang, Jingshu Mao, Wayne Xin Zhao, Cheng Wang, Peng Jiang, and Ji-Rong Wen. 2022. Feature-aware Diversified Re-ranking with Disentangled Representations for Relevant Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3327–3335.

[16] Dong Liu and Chenyang Yang. 2019. A deep reinforcement learning approach to proactive content pushing and recommendation for mobile users. *IEEE Access* 7 (2019), 83120–83136.

[17] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Ji Yang, Minmin Chen, Jiaxi Tang, Lichan Hong, and Ed H Chi. 2020. Off-policy learning in two-stage recommender systems. In *Proceedings of The Web Conference 2020*. 463–473.

[18] Shamim Nemati, Mohammad M Ghassemi, and Gari D Clifford. 2016. Optimal medication dosing from suboptimal clinical examples: A deep reinforcement learning approach. In *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2978–2981.

[19] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*. PMLR, 2778–2787.

[20] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[21] Reuven Y Rubinstein and Dirk P Kroese. 2004. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Vol. 133. Springer.

[22] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.

[23] Richard S Sutton, Andrew G Barto, et al. 1998. Introduction to reinforcement learning. (1998).

[24] Jiayin Wang, Weizhi Ma, Jiayu Li, Hongyu Lu, Min Zhang, Biao Li, Yiqun Liu, Peng Jiang, and Shaoping Ma. 2022. Make Fairness More Fair: Fair Item Utility Estimation and Exposure Re-Distribution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1868–1877.

[25] Yuyan Wang, Mohit Sharma, Can Xu, Sriraj Badam, Qian Sun, Lee Richardson, Lisa Chung, Ed H Chi, and Minmin Chen. 2022. Surrogate for Long-Term User Experience in Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4100–4109.

[26] Don R Wilhelmsen. 1974. A Markov inequality in several dimensions. *Journal of Approximation Theory* 11, 3 (1974), 216–220.

[27] Qingyun Wu, Hongning Wang, Liangjie Hong, and Yue Shi. 2017. Returning is believing: Optimizing long-term user engagement in recommender systems. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1927–1936.

[28] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. 2019. Reinforcement knowledge graph reasoning for explainable recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 285–294.

[29] Ruohan Zhan, Changhua Pei, Qiang Su, Jianfeng Wen, Xueliang Wang, Guanyu Mu, Dong Zheng, Peng Jiang, and Kun Gai. 2022. Deconfounding Duration Bias in Watch-time Prediction for Video Recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4472–4481.

[30] Qihua Zhang, Junning Liu, Yuzhuo Dai, Yiyan Qi, Yifan Yuan, Kunlun Zheng, Fan Huang, and Xianfeng Tan. 2022. Multi-Task Fusion via Reinforcement Learning for Long-Term User Satisfaction in Recommender Systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4510–4520.

[31] 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 & Data Mining*. 1040–1048.

[32] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2017. Deep reinforcement learning for list-wise recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[33] Lixin Zou, Long Xia, Zhuoye Ding, Jiaxing Song, Weidong Liu, and Dawei Yin. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2810–2818.