Do Not Wait: Learning Re-Ranking Model Without User Feedback At Serving Time in E-Commerce

Yuan Wang*†
Zhiyu Li*
Alibaba Group
Hangzhou, Zhejiang, China
wy175696@alibaba-inc.com
tuanyu.lzy@alibaba-inc.com

Changshuo Zhang
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
lyingcs@ruc.edu.cn

Sirui Chen
School of Information, Renmin University of China
Beijing, China
chensr16@gmail.com

Xiao Zhang†
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
zhangx89@ruc.edu.cn

Jun Xu
Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
junxu@ruc.edu.cn

Quan Lin
Alibaba Group
Hangzhou, Zhejiang, China
tieyi.lq@alibaba-inc.com

ABSTRACT

Recommender systems have been widely used in e-commerce, and re-ranking models are playing an increasingly significant role in the domain, which leverages the inter-item influence and determines the final recommendation lists. Online learning methods keep updating a deployed model with the latest available samples to capture the shifting of the underlying data distribution in e-commerce. However, they depend on the availability of real user feedback, which may be delayed by hours or even days, such as item purchases, leading to a lag in model enhancement. In this paper, we propose a novel extension of online learning methods for re-ranking modeling, which we term LAST, an acronym for Learning At Serving Time. It circumvents the requirement of user feedback by using a surrogate model to provide the instructional signal needed to steer model improvement. Upon receiving an online request, LAST finds and applies a model modification on the fly before generating a recommendation result for the request. The modification is request-specific and transient. It means the modification is tailored to and only to the current request to capture the specific context of the request. After a request, the modification is discarded, which helps to prevent error propagation and stabilizes the online learning procedure since the predictions of the surrogate model may be inaccurate. Most importantly, as a complement to feedback-based online learning methods, LAST can be seamlessly integrated into existing online learning systems to create a more adaptive and responsive recommendation experience. Comprehensive experiments, both offline and online, affirm that LAST outperforms state-of-the-art re-ranking models.

CCS CONCEPTS

• **Theory of computation** → **Online algorithms**; • **Computing methodologies** → **Learning paradigms**; • **Applied computing** → **Online shopping**.

KEYWORDS

online learning, re-ranking, surrogate model, recommender system, e-commerce

ACM Reference Format:
Yuan Wang, Zhiyu Li, Changshuo Zhang, Sirui Chen, Xiao Zhang, Jun Xu, and Quan Lin. 2024. Do Not Wait: Learning Re-Ranking Model Without User Feedback At Serving Time in E-Commerce. In *18th ACM Conference on Recommender Systems (RecSys '24), October 14–18, 2024, Bari, Italy.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3640457.3688165

1 INTRODUCTION

Machine learning-based and deep learning-based recommendation models have become an integral part of e-commerce platforms, such as Taobao and Amazon. Re-ranking models [1–3, 17, 19, 22, 35] typically reside in the last stage of an industrial recommendation pipeline and directly determine the final recommendation lists. They explicitly consider the mutual influence between items and explore all permutations of candidates, which makes it challenging to train a re-ranking model. Reinforcement Learning (RL) [11, 16, 26, 29] searching algorithms and an Actor-Evaluator (AE) framework [6, 9, 25] have been proposed to automatically find the best recommendation list-generating policy, removing the burden of manually specifying the best recommendation list as the label in Supervised Learning (SL). After the deployment of a re-ranking model, online learning methods [8, 23, 30–32] can be applied to continuously update the deployed model using the latest available samples so that the model can capture real-time changes in the data distribution. As shown in Fig. 1 on the left-hand side, these updates

*Both authors contributed equally to this research.
†First corresponding author: Xiao Zhang (zhangx89@ruc.edu.cn), second corresponding author: Yuan Wang (wy175696@alibaba-inc.com)

RecSys '24, October 14–18, 2024, Bari, Italy

ACM ISBN 979-8-4007-0505-2/24/10
https://doi.org/10.1145/3640457.3688165

are directly integrated into the deployed model, accumulating over time and shaping all subsequent predictions. One fundamental limitation of these methods is that they depend on the availability of real user feedback, which, akin to product purchases in e-commerce, may come several hours or even days later and ultimately constrains the temporal effectiveness of the model. Moreover, at any given moment, all requests are served with a fixed model instance, lacking contextual adaptations. In a large e-commerce platform, a recommender system may receive tens of thousands of requests within a second. These requests have their own context, reflecting diverse user preferences. A single model may not be able to capture all the variety.

In this paper, we propose LAST, an acronym for Learning At Serving Time. It ensures continuous model optimization and fine-grained model adaptation even in situations where feedback is unattainable, as illustrated on the right-hand side in Fig. 1. It uses a surrogate evaluation model as the instructional signal to steer model refinement to ensure model freshness. LAST generates transient, contextually tailored model adjustments for each request, meticulously engineered to optimize the recommendation efficacy of each request. After each recommendation, the according modification is discarded, leaving no residual influence on the deployed model. This design helps to prevent error propagation and stabilize the online learning procedure since the predictions of the surrogate model may be inaccurate. It is also more friendly to the online engineering system, as the modification functionality can be implemented as a normal model module, requiring no upgrading of the online engineering system, with or without the support of classic online learning. Most importantly, this design ensures seamless integration of LAST with existing feedback-based online learning methods. Comprehensive experiments, both offline and online, affirm that LAST outperforms state-of-the-art re-ranking models.

The main contributions of the paper are:

(1) A new re-ranking model with unique online learning ability. LAST boosts recommendation quality by feedback-independent, transient, request-specific, engineering-friendly online model modifications. It addresses the inherent temporal and adaptation limitations of conventional online learning methods and can be combined with them to enhance user satisfaction and system outcomes.

(2) Comprehensive evaluations of the new proposal. We demonstrate the effectiveness of LAST offline with publicly available data and online in an industrial environment. We have released the experimental code to increase reproducibility.

2 PRELIMINARY

In this section, we formally introduce the problem definition of re-ranking modeling and the AE framework. Given a set of M candidate items $C = \{c_i\}_{1 \le i \le M}$, a user $u \in \mathcal{U}$, and a list reward function $R(\cdot)$, the goal of a re-ranking model is to find the optimal list L_C^* composed by items in C:

$$L_C^* = \arg\max_{L_C} R(u, L_C, y_{L_C}),$$

where each list is of length N and it is obvious that $N \le M$. y_{L_C} is the feedback of the user u to the list L_C. In e-commerce, y usually involves user engagements, such as click and purchase. $R(\cdot)$ may also consider other desirable aspects of the recommended list, such as diversity [10, 18, 28], novelty [14, 27], and fairness [12, 20, 34]. Here, we assume the existence of a single optimal list. For simplicity, we henceforth drop the subscript C. A list-generating model $G(\cdot)$, parameterized by $\boldsymbol{\theta}$, is trained to find the optimal list in a single forward execution:

$$\widehat{L^*} = G(u, C; \boldsymbol{\theta})$$

In the actor-evaluator framework, $G(\cdot)$ is called the actor. The training of the actor can be described as:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{u,C}[R(u, G(u, C; \boldsymbol{\theta}), y_{G(u,C;\boldsymbol{\theta})})],$$

where $\mathbb{E}$ means expectation. However, in the training process, the actor unavoidably generates lists that have never been shown to the user, and thus y is not available. To overcome this problem, a surrogate model $E(\cdot)$, is trained to approximate $R(\cdot)$ in the actor-evaluator framework, namely $E(u, L; \boldsymbol{\phi}) \rightarrow R(u, L, y_L)$, where $\boldsymbol{\phi}$ is the parameter of $E(\cdot)$. The surrogate model is called the evaluator, and it is trained before the actor as

$$\min_{\boldsymbol{\phi}} \mathbb{E}_{u,L}[diff(E(u, L; \boldsymbol{\phi}), R(u, L, y_L))].$$

Notably, the evaluator learns how to predict user feedback during the training process. Then, the training of the actor is to find the population-wise optimal parameter set $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \max_{\boldsymbol{\theta}} \mathbb{E}_{u,C}[E(u, G(u, C; \boldsymbol{\theta}))].$$

It is crucial to see that the whole offline training process of the actor does not directly depend on user feedback, given the evaluator $E(\cdot)$.

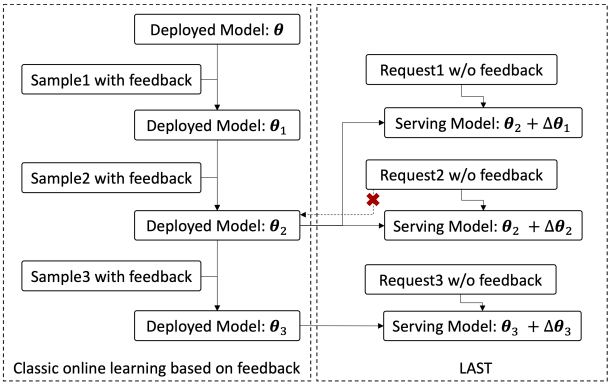

Figure 1: Classic online learning methods and our new proposal, LAST. θ means the parameter of a model. The classic methods rely on authentic user feedback, providing enduring non-request-specific updates. LAST provides transient request-specific updates with the help of a surrogate evaluation model. The two can work synergistically to create a more adaptive and responsive online serving system.

3 LAST: THE PROPOSED METHOD

Classic online learning methods rely on the real user feedback, which can be delayed by hours or even days, such as item purchases. LAST ensures continuous model optimization and fine-grained model adaptation even in the absence of user feedback. When a user

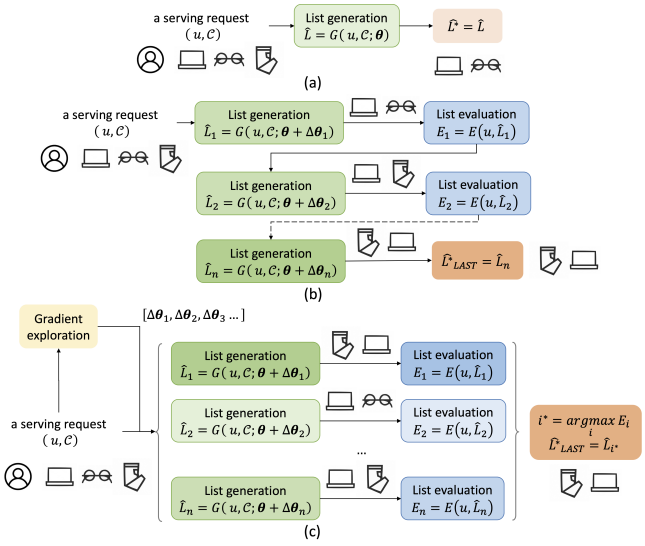

Figure 2: Online serving processes of re-ranking models. (a) The traditional re-ranking models. The model generates a recommendation list based on its fixed policy and presents the list directly to the user. (b) The cascade version of LAST. The actor interacts with the evaluator iteratively to improve its list-generating policy for higher evaluations. The list generated in the last iteration is presented to the user. (c) The parallel version of LAST. A separate gradient exploration module suggests potential model modifications. The actor tries out the suggestions, and the list with the highest evaluation is presented to the user.

request, denoted as (u, C), is received, LAST adds a request-specific disposable modification, $\Delta\boldsymbol{\theta}$, to enhance the deployed model:

$$\widehat{L^*}_{LAST} = G(u, C; \boldsymbol{\theta}^* + \Delta\boldsymbol{\theta}^*(u, C)).$$

The optimal modification $\Delta\boldsymbol{\theta}^*(u, C)$ can be obtained with the help of the evaluator:

$$\Delta\boldsymbol{\theta}^*(u, C) = \arg\max_{\Delta\boldsymbol{\theta}} E(u, G(u, C; \boldsymbol{\theta}^* + \Delta\boldsymbol{\theta})).$$

This modification is computed on the fly, tailored to the individual context of the request, capturing the unique needs and preferences of the user. After serving the current request, the modification is discarded and the model is restored to the state when the current request is received.

We introduce two versions of LAST algorithms with more details. The first is the cascade version, which finds the optimal modification $\Delta\boldsymbol{\theta}^*(u, C)$ through an iterative process as shown in Fig. 2 (b). Upon the arrival of a new user request, the cascade approach commences by generating a prediction using the currently deployed model and subsequently invokes the evaluation function, $E(\cdot)$, to assess the quality of this prediction. Thereafter, LAST endeavors to adjust the model parameters, $\boldsymbol{\theta}$, in an effort to enhance the evaluation score. The specific methods employed for updating may vary based on the application context and can range from straightforward gradient descent techniques to more intricate RL strategies. Following this initial modification of the model, LAST attempts to produce a revised list of recommendations. It proceeds to engage in a cycle of prediction, evaluation, and parameter updating until a predetermined stopping criterion is satisfied. This criterion can be defined as a minimal change in evaluation per iteration or a maximum number of iterations. The recommendation list produced in the final iteration is presented to the user. Although the cascade version of LAST can be highly effective, its iterative nature has the potential to lead to a substantial increase in the system's response time. This delay may render the cascade variant less suitable for online serving systems, where a rapid response is crucial.

The parallel version of LAST offers enhanced efficiency compared to its cascade counterpart, as shown in Fig. 2 (c). It introduces a new gradient exploration module that suggests potential model modifications. After receiving the modification suggestions, the generator attempts the suggestions, and $E(\cdot)$ estimates their effectiveness. The list with the highest evaluation is presented to the user. Algorithm. 1 shows a more concrete implementation. We explore two very special gradient directions: the one increasing the generating probability of the current list and its opposite. It is interesting that finding these directions does not involve the execution of $E(\cdot)$. Mirroring the principles of offline training, we enhance the probability of list generation when a high reward is obtained, and conversely, lower it when the reward is minor. The value obtained from $E(\cdot)$ is instrumental in calibrating the magnitude of the updates, but not the direction. The list generating probability $P(L)$ can be easily derived from a generative actor. The gradient is normalized with respect to the magnitude of $\boldsymbol{\theta}$. The normalized gradient and a set of manually specified step sizes are utilized to provide the model modification suggestions. The partial gradient of $E(\cdot)$ with respect to $\boldsymbol{\theta}$ is not directly used because $E(\cdot)$ may not be a function of $\boldsymbol{\theta}$ and thus the partial gradient does not exist. Modern neural network models can have billions of parameters. It is not necessary or efficient to modify all of them for a single request. A more feasible solution is to modify only a key subset of $\boldsymbol{\theta}$.

Algorithm 1 LAST, the parallel version implementation.

Require: a user u; a candidate item set C; a deployed actor model $G(\cdot; \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$; a list evaluation function $E(\cdot)$; a function $P(L)$ indicating the probability of G generating list L; a list of step size $[\eta_1, \eta_2, ...]$; a constant factor α for gradient normalization

Ensure: a predicted optimal recommendation list $\widehat{L^*}_{LAST}$ for u, which is composed by items in C

$\hat{L} \leftarrow G(u, C; \boldsymbol{\theta})$ {run the prediction model}
$\boldsymbol{g}_{\boldsymbol{\theta}} \leftarrow \frac{\partial P(\hat{L})}{\partial \boldsymbol{\theta}}$ {calculate the partial derivative of P with respect to $\boldsymbol{\theta}$}
$\Delta\boldsymbol{\theta} \leftarrow \alpha \frac{|\boldsymbol{\theta}|}{|\boldsymbol{g}_{\boldsymbol{\theta}}|} \boldsymbol{g}_{\boldsymbol{\theta}}$ {normalize the gradient}
for η in $[\eta_1, \eta_2, ...]$ **do**
  $\hat{L}_\eta \leftarrow G(u, C; \boldsymbol{\theta} + \eta\Delta\boldsymbol{\theta})$
  $E_\eta \leftarrow E(u, \hat{L}_\eta)$
end for
$\eta^* = \text{argmax}_\eta E_\eta$
return $\widehat{L^*}_{LAST} = L_{\eta^*}$

4 EXPERIMENTS

4.1 Offline Experiments

We conduct our offline experiments on the benchmark LibRerank[1] with the public recommendation dataset Ad2[2]. The source code has been released[3]. The original Ad dataset records 1 million users and 26 million ad display/click logs, with 8 user profiles (e.g., id, age, and occupation), 6 item features (e.g., id, campaign, and brand). LibRerank transformed the records of each user into ranking lists according to the timestamp of the user browsing the advertisement. The final Ad dataset contains 349,404 items and 483,049 lists. We chose a wide range of representative and state-of-the-art re-ranking methods as baselines, including GSF [2], DLCM [1], PRM [22], SetRank [21], EGR [25], and CMR [5]. CMR can work in two modes. In the greedy mode, it picks the item with the largest selection probability in each step and generates only one recommendation list for each request. In the sampling mode, it generates multiple recommendation lists using Thompson Sampling, and the list with the highest evaluation score is presented to a user. We implement the parallel version of the LAST. The backbone model structure of the actor and the entire structure of the evaluator are the same as CMR. LAST adds the gradient exploration module to the actor, which only functions in online serving. Offline training is the same for LAST and CMR, including the training objectives and procedure. Our offline experiments aim to answer two key questions: (i) Does LAST outperform the latest re-ranking models? and (ii) How do hyper-parameters impact the effectiveness of LAST?

4.1.1 Performance Analysis. In the first offline experiment, we carry out a wide comparison between re-ranking methods, AE and non-AE ones. To make the comparison fair and appreciable, we use popular metrics such as Normalized Discounted Cumulative Gain (NDCG) for evaluation. In this case, an item not initially exposed to users is given a negative pseudo-label. A more subtle assumption is that user feedback to items stays the same while a recommendation list has been re-arranged, which does not seem to be very plausible from the perspective of re-ranking modeling. However, these assumptions constitute a simple experiment evaluation protocol widely used in related works. The non-AE approaches obtain the final recommendation list by ranking candidates according to the engagement probability prediction of each item from high to low. The AE re-ranking models use NDCG as the evaluator.

Table. 1 summarizes the results. It is interesting to see that the non-AE methods, including GSF, DLCM, SetRank, and PRM, outperform the AE methods, i.e. EGR and CMR(Greedy). These AE methods use a single-list strategy, which means the actor will only generate a single recommendation list, by picking the item with the largest selection probability in each step. We think it is because "ranking the items according to the predicted user engagement probabilities" is a very strong prior, it significantly reduced the modeling difficulty. RL searching algorithms do not know this prior and thus converge to a worse local optimal. When it comes to AE re-ranking models with the multi-list strategy including CMR(Sampling) and LAST, they beat other baselines by a large margin. It reflects the simple fact that more trials can always lead to better results, in probability. The margins seem surprisingly large. This is because MAP and NDCG are discrete functions and heavily emphasize the top lists. For example, the NDCG value of the perfect list "1, 0, 0, 0, 0, 0, 0, 0, 0, 0" is 1 and 0.63 for a close list "0, 1, 0, 0, 0, 0, 0, 0, 0, 0". The former is nearly 59 percent higher than the latter, which appears to be exaggerated. Here 1 in the list means a relevant item and 0 means an irrelevant one. LAST performs significantly better than all other baselines, which supports the benefit of serving time adaptation.

Table 1: Offline comparison of re-ranking models. The AE re-ranking models use the NDCG metric as the evaluator. The best results is bolded and the runner-up is underlined. * indicates that the improvement over the best baseline is statistically significant (p-value < 0.05).

Method	map@5	map@10	ndcg@5	ndcg@10
GSF	0.6038	0.6074	0.6838	0.6984
DLCM	0.6169	0.6201	0.6948	0.7080
SetRank	0.6084	0.6122	0.6878	0.7020
PRM	0.6176	0.6209	0.6950	0.7087
EGR	0.6010	0.6047	0.6822	0.6964
CMR(Greedy)	0.6032	0.6067	0.6838	0.6979
CMR(Sampling)	<u>0.6559</u>	<u>0.6587</u>	<u>0.7243</u>	<u>0.7370</u>
LAST	**0.6623***	**0.6652***	**0.7289***	**0.7418***

In the second offline experiment, we train the AE re-ranking models with a pre-trained evaluator, which is the standard practice of the AE framework, leading to better online performance experimentally. In this experiment, the model structure of the evaluator is the same as in CMR, and it tries to predict the click probability of each item. We use evaluator@N to indicate the quality of the recommendation list, which represents the item-wise average click probability of the top N items. The results in Table. 2 clearly show the advantage of re-ranking models with the multi-list strategy over the ones with the single-list strategy, and the advantage of LAST over baselines. The relative improvements appear to be more reasonable compared to the first offline experiment, which is more likely to indicate the improvement online.

Table 2: Offline comparison of re-ranking models. The AE re-ranking models use a pre-trained evaluator to predict user engagement. The best results is bolded and the runner-up is underlined. * indicates that the improvement over the best baseline is statistically significant (p-value < 0.05).

Method	evaluator@5	evaluator@10
EGR	0.3215	0.3086
CMR(Greedy)	0.3259	0.3131
CMR(Sampling)	<u>0.3395</u>	<u>0.3267</u>
LAST	**0.3438***	**0.3310***

[1]https://github.com/LibRerank-Community/LibRerank
[2]https://tianchi.aliyun.com/dataset/56
[3]https://github.com/lyingCS/LAST

4.1.2 Hyper-parameters Analysis. We evaluate the impact of two core hyper-parameters of LAST in the third offline experiment. Fig. 3(a) shows the influence of the length of the step size lists $[\eta_1, \eta_2, ...]$, where each η generates a recommendation list. We can see that more list trials always lead to higher evaluation scores and LAST is clearly more effective than CMR(Sampling). Fig. 3(b) depicts the influence of the normalization factor α. When α is too small, the modification is not strong enough to effectively change the recommendation lists to increase the evaluation score; when α goes too big, the direction indicated by the local gradient becomes unreliable. Both lead to a suppressed LAST improvement. The performance of LAST reaches its peak when α is 1%.

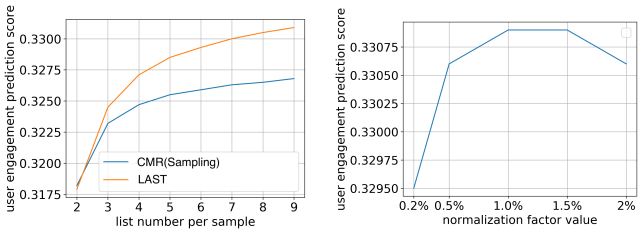

Figure 3: The impact of hyper-parameters on LAST.

4.2 Online Experiments

We conduct online experiments on the "Subscribe" scene in the Taobao App, a leading e-commerce platform in China. Its main entrance is the "Subscribe" button on top of Taobao's main landing page and it is a stream of various elements including items, posters, videos, etc. 2,003,565 users were involved in the 7-day long experiments. In the online experiment, we aim to improve the purchase number per user, meanwhile maintaining the user interaction frequency, indicated by the click number per user. CMR(Sampling) is the baseline method, which serves the main traffic of our online scene and LAST is the experiment method. The main difference between the two is that LAST uses online request-adaptive modifications. Other aspects, such as the model structure, the offline training process, and the exploration number of lists during online serving are the same between CMR(Sampling) and LAST. We do not have classic online learning methods deployed at this moment because of engineering limitations. We leave the combination of classic online learning methods and LAST in further study. The results from the online A/B testing are summarized in Table 3, where the superiority of LAST is evident. By explicitly adapting its model parameters in real-time to each incoming request, LAST is able to achieve a higher conversion rate, leading to more purchases, without detracting from user interaction. With our current implementation, the average online response time is 24ms for CMR(sampling) and 32ms for LAST. In daily use, it is acceptable, but during promotional periods, acceleration is desirable which we are working on.

5 DISCUSSION

Superficially from the problem definition, it is easy to see that $\widehat{L^*}_{LAST}$ can achieve a higher evaluator score than $\widehat{L^*}$ for each request. The underlying question is: is there any new information we can utilize by only knowing the request without user feedback? Our answer is yes. Fundamentally, as long as two random variables are not independent, like a request and its optimal parameter set, one can get information about one random variable by knowing the value of another. In the semi-supervised learning [13, 15, 33] community, it has been widely accepted that unlabeled data can be beneficial. In fact, the whole idea of semi-supervised learning is rooted in it. Another piece of evidence is documented in the book [4] where it is proved that the Vovk–Azoury–Warmuth forecaster can achieve an improved logarithmic regret bound using the input information at the last time point without the target value. In LAST, we use the information contained in the new request for model updating.

LAST appears to be prone to overfitting since the prediction of the evaluator is not noise-free and, as a result, may mislead model updating. In this paper, we demonstrate its effectiveness with online experiments. Robust estimation techniques, such as Double Q-learning [7, 24], can be applied to alleviate this problem and may further improve the performance of LAST.

Table 3: Online A/B test results, LAST over CMR(Sampling).

Metric	Relative improvement
click number per user	0.08%
purchase number per user	2.08%

6 CONCLUSION

We propose a novel re-ranking model in e-commerce with a unique online learning ability. Distinct from existing methods, it can achieve effective model online learning without waiting for real user feedback, by using supervision signals provided by a surrogate model. Deliberated algorithmic designs are introduced to fulfill its full potential. It introduces request-specific modifications to maximize model adaptation to the context of each request. It discards the modifications after each recommendation to stabilize the training procedure. These designs ensure LAST can work in harmony with existing online learning systems while providing improved temporal effectiveness and incremental adaptation. Through extensive offline and online experiments, we demonstrate the strength of the new proposed model.

ACKNOWLEDGMENTS

This work was funded by the National Science and Technology Major Project (2022ZD0114802), the National Natural Science Foundation of China (No. 62376275, 62377044), Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. Supported by fund for building world-class universities (disciplines) of Renmin University of China. Supported by Public Computing Cloud, Renmin University of China. Supported by the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China (23XNKJ13).

REFERENCES

[1] Qingyao Ai, Keping Bi, Jiafeng Guo, and W Bruce Croft. 2018. Learning a deep listwise context model for ranking refinement. In *The 41st international ACM SIGIR conference on research & development in information retrieval*. 135–144.

[2] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. 2019. Learning groupwise multivariate scoring functions using deep neural networks. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval*. 85–92.

[3] Irwan Bello, Sayali Kulkarni, Sagar Jain, Craig Boutilier, Ed Chi, Elad Eban, Xiyang Luo, Alan Mackey, and Ofer Meshi. 2018. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019* (2018).

[4] Nicolò Cesa-Bianchi and Gabor Lugosi. 2006. *Prediction, learning, and games*. Cambridge University Press.

[5] Sirui Chen, Yuan Wang, Zijing Wen, Zhiyu Li, Changshuo Zhang, Xiao Zhang, Quan Lin, Cheng Zhu, and Jun Xu. 2023. Controllable Multi-Objective Re-ranking with Policy Hypernetworks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3855–3864.

[6] Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q Zhu. 2019. Exact-k recommendation via maximal clique optimization. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 617–626.

[7] Hado Hasselt. 2010. Double Q-learning. *Advances in neural information processing systems* 23 (2010).

[8] Elad Hazan. 2016. Introduction to online convex optimization. *Foundations and Trends® in Optimization* 2, 3-4 (2016), 157–325.

[9] Guangda Huzhang, Zhenjia Pang, Yongqing Gao, Yawen Liu, Weijie Shen, Wen-Ji Zhou, Qing Da, Anxiang Zeng, Han Yu, Yang Yu, et al. 2021. AliExpress Learning-To-Rank: Maximizing online model performance without going online. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[10] Zhengbao Jiang, Ji-Rong Wen, Zhicheng Dou, Wayne Xin Zhao, Jian-Yun Nie, and Ming Yue. 2017. Learning to diversify search results via subtopic attention. In *Proceedings of the 40th international ACM SIGIR Conference on Research and Development in Information Retrieval*. 545–554.

[11] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research* 4 (1996), 237–285.

[12] Chen Karako and Putra Manggala. 2018. Using image fairness representations in diversity-based re-ranking for recommendations. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*. 23–28.

[13] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. *Advances in neural information processing systems* 27 (2014).

[14] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.

[15] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3. Atlanta, 896.

[16] Yuxi Li. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* (2017).

[17] Yi Li, Jieming Zhu, Weiwen Liu, Liangcai Su, Guohao Cai, Qi Zhang, Ruiming Tang, Xi Xiao, and Xiuqiang He. 2022. Pear: Personalized re-ranking with contextualized transformer for recommendation. In *Companion Proceedings of the Web Conference 2022*. 62–66.

[18] Jiongnan Liu, Zhicheng Dou, Xiaojie Wang, Shuqi Lu, and Ji-Rong Wen. 2020. DVGAN: a minimax game for search result diversification combining explicit and implicit features. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 479–488.

[19] Mohammadmehdi Naghiaei, Hossein A Rahmani, and Yashar Deldjoo. 2022. Cpfair: Personalized consumer and producer fairness re-ranking for recommender systems. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 770–779.

[20] Harrie Oosterhuis. 2021. Computationally efficient optimization of plackett-luce ranking models for relevance and fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1023–1032.

[21] Liang Pang, Jun Xu, Qingyao Ai, Yanyan Lan, Xueqi Cheng, and Jirong Wen. 2020. Setrank: Learning a permutation-invariant ranking model for information retrieval. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 499–508.

[22] 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.

[23] Doyen Sahoo, Quang Pham, Jing Lu, and Steven C. H. Hoi. 2018. Online Deep Learning: Learning Deep Neural Networks on the Fly. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2660–2666.

[24] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 30.

[25] Fan Wang, Xiaomin Fang, Lihang Liu, Yaxue Chen, Jiucheng Tao, Zhiming Peng, Cihang Jin, and Hao Tian. 2019. Sequential evaluation and generation framework for combinatorial recommender system. *arXiv preprint arXiv:1902.00245* (2019).

[26] Marco A Wiering and Martijn Van Otterlo. 2012. Reinforcement learning. *Adaptation, learning, and optimization* 12, 3 (2012), 729.

[27] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2016. Modeling document novelty with neural tensor network for search result diversification. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. 395–404.

[28] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, Wei Zeng, and Xueqi Cheng. 2017. Adapting Markov decision process for search result diversification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 535–544.

[29] Changshuo Zhang, Sirui Chen, Xiao Zhang, Sunhao Dai, Weijie Yu, and Jun Xu. 2024. Reinforcing Long-Term Performance in Recommender Systems with User-Oriented Exploration Policy. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1850–1860.

[30] Xiao Zhang, Haonan Jia, Hanjing Su, Wenhan Wang, Jun Xu, and Ji-Rong Wen. 2021. Counterfactual reward modification for streaming recommendation with delayed feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 41–50.

[31] Xiao Zhang and Shizhong Liao. 2019. Incremental randomized sketching for online kernel learning. In *Proceedings of the 36th International Conference on Machine Learning*. 7394–7403.

[32] Xiao Zhang, Yun Liao, and Shizhong Liao. 2019. A survey on online kernel selection for online kernel learning. *WIREs Data Mining and Knowledge Discovery* 9, 2 (2019), e1295. https://doi.org/10.1002/widm.1295

[33] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. 2003. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*. 912–919.

[34] Ziwei Zhu, Jingu Kim, Trung Nguyen, Aish Fenton, and James Caverlee. 2021. Fairness among new items in cold start recommender systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 767–776.

[35] Tao Zhuang, Wenwu Ou, and Zhirong Wang. 2018. Globally optimized mutual influence aware ranking in e-commerce search. *arXiv preprint arXiv:1805.08524* (2018).