ReAlign: Reward-Guided Preference Alignment for Generative Re-Ranking

YaChen Yan
yachen_yan@intuit.com
Intuit Credit Karma
San Francisco, California, USA

Liubo Li
liubo_li@intuit.com
Intuit Credit Karma
San Francisco, California, USA

ABSTRACT

Re-ranking refines the upstream ranked list to maximize listwise user utility. Existing two-stage generator-evaluator methods decouple list generation from list evaluation, leaving the generator without direct access to the listwise utility signal, while scoring-based one-stage methods score items under a fixed upstream permutation context without explicitly searching the permutation space. We propose **ReAlign**, a reward-guided preference alignment framework that couples a listwise reward model with a candidate-aware autoregressive generator. The reward model decomposes the listwise reward into position-wise contributions, modeling user exposure through a survival-based exposure module and item value through per-objective heads for heterogeneous engagement signals. The generator is trained in two stages: supervised pre-training on logged exposure lists to establish a relevance prior, followed by reward-guided post-training that aligns the generator with the reward model through complementary distribution and ranking alignment over within-pool standardized rewards. Extensive offline experiments show that ReAlign consistently outperforms existing state-of-the-art baselines.

CCS CONCEPTS

• **Computing methodologies**; • **Machine learning**; • **Machine learning approaches**; • **Neural networks**;

KEYWORDS

Recommender Systems, Generative Recommendation, Re-ranking, Preference Optimization

ACM Reference Format:

YaChen Yan and Liubo Li. 2025. ReAlign: Reward-Guided Preference Alignment for Generative Re-Ranking. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Re-ranking is the final stage of a modern recommender system, where an upstream ranking module produces a small candidate set and the re-ranker arranges these candidates into an ordered list shown to the user. Although the candidate set is short, the number of possible orderings grows factorially, and the user utility of a list depends on intra-list interactions such as competition, complementarity, and position-dependent exposure. Effectively searching this combinatorial space, while modeling how users actually consume an ordered list, is the central challenge of re-ranking.

Despite substantial progress, two gaps remain. First, listwise reward estimation is often coarse: existing evaluators commonly collapse user exposure and engagement into a single scalar, ignoring the cascading nature of user browsing and the heterogeneity of engagement objectives that arise in real systems. A re-ranker that relies on such a coarse signal cannot easily distinguish lists that differ only in deep-position behavior or in the trade-off across objectives. Second, even with a high-quality listwise signal, transferring that signal to a generator that explicitly searches the permutation space is non-trivial. Two-stage generator-evaluator pipelines decouple generation from evaluation and limit end-to-end optimization, while preference-based generator training is sensitive to the absolute scale of the reward and is known to be unstable when applied from random initialization in cold-start settings [12].

We propose **ReAlign**, a reward-guided preference alignment framework that addresses both gaps with a tightly coupled reward model and generator. The reward model decomposes the listwise reward into position-wise contributions, modeling user exposure as a discrete-time survival process and item value through multiple per-objective heads with uncertainty-based weighting. The generator is trained in two stages: a supervised pre-training stage that imitates logged exposure lists to establish a stable relevance prior, followed by a reward-guided post-training stage that aligns the generator with reward-induced listwise preferences over a diverse list pool. The post-training objective combines distribution alignment with a ranking alignment loss, both operating on within-pool standardized rewards to remain robust to the absolute scale and bounded biases of the reward.

Our contributions are summarized as follows.

- We propose ReAlign, a reward-guided preference alignment framework for generative re-ranking that combines a listwise reward model with a candidate-aware autoregressive generator under a unified training pipeline.
- We design a decomposable listwise reward model that jointly models user exposure via a discrete-time survival process and item value via multi-objective heads with uncertainty-based weighting, yielding a calibrated and interpretable listwise reward.
- We develop a reward-guided post-training objective that combines distribution alignment and ranking alignment over a diverse list pool, with within-pool reward standardization

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

that makes both terms robust to reward scale and bounded biases.

2 RELATED WORK

2.1 Re-ranking in Recommender Systems

Re-ranking models aim to refine an initial ranked list by modeling item interactions and optimizing list-level user utility. Existing approaches can be broadly categorized into one-stage and two-stage methods.

One-stage methods directly model the target ranking in a single framework. Early works focus on listwise scoring with context modeling, where DLCM and PRM incorporate list context into deep models to refine item scores [1, 15], and SetRank introduces permutation-invariant representations for set-level structure [14]. Some approaches directly model permutations as structured outputs emphasizing global optimization [7, 29], while more recent work extends this to generative re-ranking in autoregressive or non-autoregressive manners [8, 17].

Two-stage methods decompose re-ranking into a generator-evaluator framework. Seq2Slate generates slates via sequential decoding [2], and PIER combines permutation-level generation with interest-aware evaluation [18]. Recent works further improve candidate diversity and evaluation quality through multi-generator construction [22], neighbor-aware generation [20], and unified generative frameworks [12]. Newer studies also explore better optimization targets for the evaluator, including list-level value estimation and group-relative optimization [26, 27].

2.2 Generative Recommendation and Preference Optimization

Generative recommendation reframes item retrieval as sequence decoding. TIGER [16] introduced semantic item IDs for autoregressive generative retrieval, and HSTU [24] scaled this paradigm to trillion parameters with LLM-like scaling behavior. Industrial deployments have followed [9, 10, 23, 25], and several works pursue end-to-end unification of retrieval and ranking: OneRec [6, 28] collapses the multi-stage pipeline into a single generative model; OneSearch [4] applies this to e-commerce search; and GR4AD [21] co-designs architecture, training, and serving for large-scale advertising.

Aligning generative recommenders with user preferences draws from RLHF and direct preference optimization. A related line of work extends preference contrast to listwise settings: PRO [19] decomposes a reward-ranked list into iterative one-vs-rest softmax classifications, and LiPO [13] connects preference optimization to learning-to-rank, improving alignment calibration over standard DPO-style objectives. In the recommendation context, OneRec [6, 28] applies iterative DPO alignment with online feedback; OneSearch [4] employs a listwise DPO objective with adaptive reward-weighted ranking; and GR4AD [21] proposes RSPO, an NDCG-weighted DPO variant that optimizes ranking quality beyond short-term click feedback. This line of work highlights that the choice of preference optimization objective plays a central role in enabling generative recommenders to align with real-world objectives.

3 METHODOLOGY

We describe ReAlign in detail, covering the framework overview, the shared user and item encoders, the listwise reward model, the candidate-aware generator with its two-stage training pipeline, and the overall training and inference procedure.

3.1 Framework Overview

Our framework, **ReAlign** (Re-ranking via reward-guided alignment), consists of two main components: a reward model and a generator. The reward model acts as an evaluator that estimates the expected utility of an ordered list by jointly modeling exposure and item value under user browsing behavior. The generator is a candidate-aware encoder–decoder model that autoregressively constructs ordered lists from the upstream candidate set. We train the framework in a staged manner. First, the reward model is learned from logged recommendation data with observed exposure and engagement signals. Then, the generator is trained in two stages: supervised pre-training on logged exposure lists, followed by reward-guided post-training through preference alignment. At serving time, the generator performs autoregressive decoding over the candidate set to produce the final re-ranked list.

3.2 Problem Formulation

We consider a multi-stage recommendation setting with a user space $\mathcal{U}$ and an item space $\mathcal{I}$. Let $u \in \mathcal{U}$ denote a user, and let $I \in \mathcal{I}$ denote an item.

Given a user u, an upstream ranking stage produces a set of candidate items $X = \{I_1, I_2, \ldots, I_m\}$, where $I_i \in \mathcal{I}$.

The re-ranking task aims to generate an ordered list of items $L = (I_{l_1}, I_{l_2}, \ldots, I_{l_n})$, where $L \subseteq X$ and $n \ll m$.

We model re-ranking as a sequence generation problem. Specifically, we define a conditional distribution $P(L \mid u, X)$, where L belongs to the space of ordered subsets of X. The ordered list is generated in an autoregressive manner:

$$P(L \mid u, X) = \prod_{t=1}^{n} P(I_{l_t} \mid u, X, I_{l_{<t}}). \tag{1}$$

To evaluate and guide the generation of ordered lists, we define a reward function $r(u, L)$ over ordered lists conditioned on the user, which assigns a scalar value to each ordered list L for user u.

The objective of the re-ranking problem is to learn a conditional distribution $P(L \mid u, X)$ that produces ordered lists L maximizing the reward $r(u, L)$.

3.3 User and Item Encoders

A key component of our framework is a shared representation scheme that maps users and items into embedding tokens for downstream modeling. We introduce a user encoder and an item encoder to transform each user and each item into dense representations, which can be viewed as user and item tokens. These tokens serve as a common interface for subsequent modules. In the following, we first describe the user encoder and then present the item encoder.

3.3.1 User Encoder. The user encoder transforms each user into a fixed set of user tokens that capture different aspects of user behavior. As illustrated in Figure 1, it consists of three components

that process user static features, long-term interaction sequences, and short-term interaction sequences, respectively.

For user static features, we apply a linear transformation to map the input features into a fixed-dimensional representation, which is then split into multiple tokens. For the long-term interaction sequence, we adopt a multi-interest extraction layer that extracts multiple interest vectors via a softmax-based attention mechanism [3], each corresponding to a distinct latent interest. For the short-term interaction sequence, we use the most recent N interactions as tokens to capture the user's immediate interests. The tokens from these three components are concatenated to form the final set of user tokens, followed by RMS normalization to stabilize the representation.

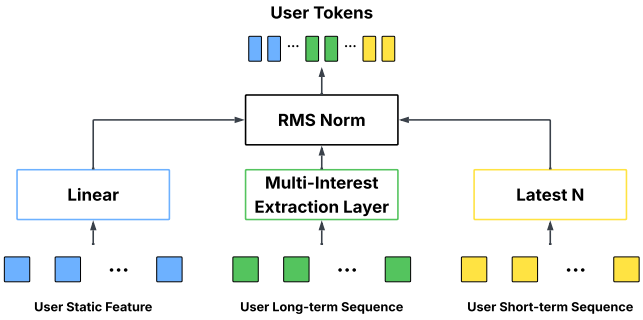

Figure 1: The user encoder maps heterogeneous user signals into a fixed set of user tokens, which are normalized to form a unified representation for downstream modules.

3.3.2 Item Encoder. We adopt a lightweight design for the item encoder that combines item identity with side information. For each item, we learn a trainable ID embedding to capture collaborative signals, and encode its side information (e.g., category, price, and other metadata) into a dense representation via a linear projection. The two components are combined through element-wise addition, forming a unified item token that reflects both identity and feature semantics. Finally, we apply RMS normalization to stabilize the representation.

3.4 Reward Model

We introduce a reward model that plays the role of a listwise evaluator in the re-ranking stage, assigning a scalar reward to each candidate list for a given user. We use the term *reward model* to emphasize its role in guiding downstream optimization, while it is conceptually equivalent to an evaluator in prior re-ranking frameworks.

Given a user u and an ordered list of items $L = (I_{l_1}, \ldots, I_{l_n})$, the reward model is defined as $r_\phi(u, L)$, where ϕ denotes the model parameters.

3.4.1 Listwise Reward Decomposition. Rather than directly modeling the reward of the entire list, we decompose the listwise reward into position-wise contributions under the user browsing process:

$$r_\phi(u, L) = \sum_{t=1}^{n} \underbrace{p_\phi(t \mid u, L)}_{\text{reaching probability}} \cdot \underbrace{v_\phi(u, I_{l_t})}_{\text{item value}}, \tag{2}$$

where $p_\phi(t \mid u, L)$ denotes the probability that the user reaches and observes the item at position t, and $v_\phi(u, I_{l_t})$ denotes the expected value of the item conditional on being observed. Each item contributes to the listwise reward proportionally to the probability that the user actually sees it, yielding the expected utility under the browsing process. This decomposition naturally motivates modeling reaching probability and item value as two coupled components within a unified architecture.

3.4.2 Reward Model Architecture. To jointly model reaching probability and item value, we employ a unified transformer-based architecture over the ordered list, as illustrated in Figure 2. This architecture takes ranked item tokens and user tokens as input and produces context-aware ranked item tokens for all positions.

Specifically, before being processed by the transformer, each ranked item token is summed with a learned position embedding corresponding to its rank in the list. Position is a critical signal in re-ranking, as it directly governs user exposure and shapes how items are perceived in context. The position-augmented item tokens are then processed by a stack of transformer layers, where each layer consists of cross-attention to user tokens, self-attention over the ranked item sequence, and a feed-forward network, each sub-layer equipped with RMS normalization and residual connections. The cross-attention module injects user-specific preferences into each position, the self-attention module captures listwise dependencies among items (e.g., competition, redundancy, and complementarity), and the feed-forward network introduces position-wise nonlinearity to refine the contextualized representations.

The resulting context-aware ranked item tokens are then fed into two types of task-specific heads:

- a **survival head** predicts the position-wise hazards $h_\phi(t \mid u, L)$, from which reaching probabilities can be derived,
- K **value heads**, one per objective, each predicting a per-objective value estimate $\hat{v}_\phi^{(k)}(u, I_{l_t})$ that is later aggregated into the item value $v_\phi(u, I_{l_t})$.

This design disentangles reaching probability from item value while allowing joint optimization through shared representations.

3.4.3 Exposure Modeling. We model the user's exposure at each position via a *reaching probability* $p_\phi(t \mid u, L)$, defined as the probability that the user reaches and observes the item at position t. Let $E_t \in \{0, 1\}$ denote whether the item at position t is observed. We assume a top-down sequential browsing process, where users scroll through the list and may exit at any position.

Sequential Browsing Intuition. In practice, users interact with a recommendation list sequentially: at each position, a user either exits or continues scrolling to the next. This means exposure is inherently cascading: a user cannot reach position t without having chosen to continue past all preceding positions. Such behavior is naturally captured by a survival process, where each position presents a discrete risk of exit, and the probability of reaching deeper positions decreases monotonically as these risks accumulate.

Survival-Based Formulation. We formalize this intuition as a discrete-time survival process over a ranked list of n positions. Let M denote the position at which the user exits. The fundamental quantity is the **conditional hazard**:

$$h_\phi(t \mid u, L) = P(M = t \mid M \geq t,\ u, L), \tag{3}$$

the probability of exit at position t given the user has reached it. At each position, the user exits with probability $h_\phi(t)$ or continues with probability $1-h_\phi(t)$. The reaching probability is then obtained as the survival function of this process:

$$p_\phi(t \mid u, L) = \prod_{j=1}^{t-1}\big(1 - h_\phi(j \mid u, L)\big), \tag{4}$$

with $p_\phi(1 \mid u, L) = 1$ by convention, yielding a monotonically decreasing reaching probability with depth that recovers the cascading exposure structure.

Training Objective. Let $C = \min(M, n)$ denote the censoring position, and $\delta = \mathbf{1}[M \leq n]$ the event indicator. The likelihood of an observed browsing trajectory under the discrete-time survival process is $P(C, \delta \mid u, L) = h_\phi(C \mid u, L)^\delta \prod_{t=1}^{C-\delta}\big(1 - h_\phi(t \mid u, L)\big)$, yielding the negative log-likelihood

$$\mathcal{L}_{\text{exp}} = -\,\delta \log h_\phi(C \mid u, L) \;-\; \sum_{t=1}^{C-\delta} \log\big(1 - h_\phi(t \mid u, L)\big). \tag{5}$$

Equivalently, defining per-position exit indicators $y_t = \delta\,\mathbf{1}[t = C]$ over the reached positions $t \leq C$, $\mathcal{L}_{\text{exp}}$ is the binary cross-entropy between y_t and $h_\phi(t \mid u, L)$; right-censored trajectories ($\delta = 0$) contribute survival terms at every reached position and no event term.

3.4.4 Value Modeling. In practice, recommendation systems typically involve multiple engagement objectives, such as clicks, watch time, likes, and shares. Rather than collapsing these signals into a single scalar through a fixed weighted combination, we adopt a multi-objective framework that models each objective independently with a dedicated prediction head, improving both interpretability and optimization.

Formally, let $O = \{1, \ldots, K\}$ denote the set of K objectives. For each objective $k \in O$, let $Y_t^{(k)} \geq 0$ denote the observed utility for objective k at position t, which is sparse and non-negative (e.g., zero for unengaged impressions and positive for engaged ones). The aggregated item value is defined as a weighted sum of per-objective value estimates:

$$v_\phi(u, I_{l_t}) = \sum_{k=1}^{K} w_k \cdot \hat{v}_\phi^{(k)}(u, I_{l_t}), \tag{6}$$

where $\hat{v}_\phi^{(k)}(u, I_{l_t})$ is the per-objective value estimate for objective k, and w_k is the manually specified business importance weight that reflects the relative value of each objective at inference time. These weights encode product-level preferences and are distinct from the training-time loss weights introduced later for balancing optimization across heterogeneous objectives.

Per-Objective Head and Weighted Logistic Regression. Each objective k is modeled via a dedicated value head that produces a scalar logit $f_\phi^{(k)}(u, I_{l_t})$ from the shared context-aware representation. Following Covington et al. [5], we adopt weighted logistic regression to handle sparse, non-negative utility labels. Every observed impression contributes a unit negative term, and an impression with $Y_t^{(k)} > 0$ additionally contributes a positive term weighted by $Y_t^{(k)}$, so exponentiating the logit recovers a calibrated estimate of the expected utility. At inference time, the per-objective value estimate is therefore obtained by exponentiating the logit:

$$\hat{v}_\phi^{(k)}(u, I_{l_t}) = \exp\Big(f_\phi^{(k)}(u, I_{l_t})\Big). \tag{7}$$

Per-Objective Training Loss. The per-objective loss is a weighted cross-entropy over observed positions:

$$\mathcal{L}_{\text{value}}^{(k)} = -\sum_{t:\,E_t=1} \left[Y_t^{(k)} \cdot \log\sigma\Big(f_\phi^{(k)}(u, I_{l_t})\Big) + \log\Big(1 - \sigma\Big(f_\phi^{(k)}(u, I_{l_t})\Big)\Big)\right], \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function. Positive examples with larger $Y_t^{(k)}$ contribute proportionally more to the loss, guiding the model to accurately predict the magnitude of the utility rather than merely its occurrence.

Uncertainty-Based Multi-Task Weighting. A key challenge in multi-objective learning is balancing the training losses across objectives with different scales and noise levels, independently of the business importance weights used at inference time. Following Kendall et al. [11], we adopt an uncertainty-based multi-task learning approach that automatically learns per-task loss weights by modeling the homoscedastic uncertainty of each objective. For each objective k, we introduce a learnable log-uncertainty parameter $s_k = \log\sigma_k^2$, where $\sigma_k > 0$ captures the observation noise of task k. The uncertainty-weighted loss for objective k is: $\widetilde{\mathcal{L}}_{\text{value}}^{(k)} = e^{-s_k}\mathcal{L}_{\text{value}}^{(k)} + \frac{1}{2}s_k$. The term e^{-s_k} down-weights objectives with higher uncertainty, while the regularization term $\frac{1}{2}s_k$ prevents the model from trivially driving $\sigma_k \to \infty$. The parameters $\{s_k\}_{k=1}^K$ are optimized jointly with the model parameters ϕ.

Training Objective. The overall value training objective is:

$$\mathcal{L}_{\text{value}} = \sum_{k=1}^{K} \widetilde{\mathcal{L}}_{\text{value}}^{(k)} = \sum_{k=1}^{K} \left(e^{-s_k}\mathcal{L}_{\text{value}}^{(k)} + \frac{1}{2}s_k \right). \tag{9}$$

3.4.5 Joint Training Objective. The reward model can be viewed as factorizing a joint distribution over exposure and engagement signals:

$$P(E_{1:n}, \mathbf{Y}_{1:n} \mid u, L) = P(E_{1:n} \mid u, L) \cdot P(\mathbf{Y}_{1:n} \mid E_{1:n}, u, L), \tag{10}$$

where $\mathbf{Y}_{1:n} = \{Y_t^{(k)}\}_{t=1,\ldots,n;\,k=1,\ldots,K}$ denotes the collection of per-position multi-objective engagement signals, and the reward model is trained by minimizing:

$$\mathcal{L} = \mathcal{L}_{\text{exp}} + \mathcal{L}_{\text{value}}. \tag{11}$$

At inference time, the listwise reward in Eq. (2) is computed by combining the reaching probability (Eq. (4)) with the aggregated item value (Eq. (6)), yielding a list-level utility that jointly reflects exposure probability, item value and inter-item dependencies induced by the ordering.

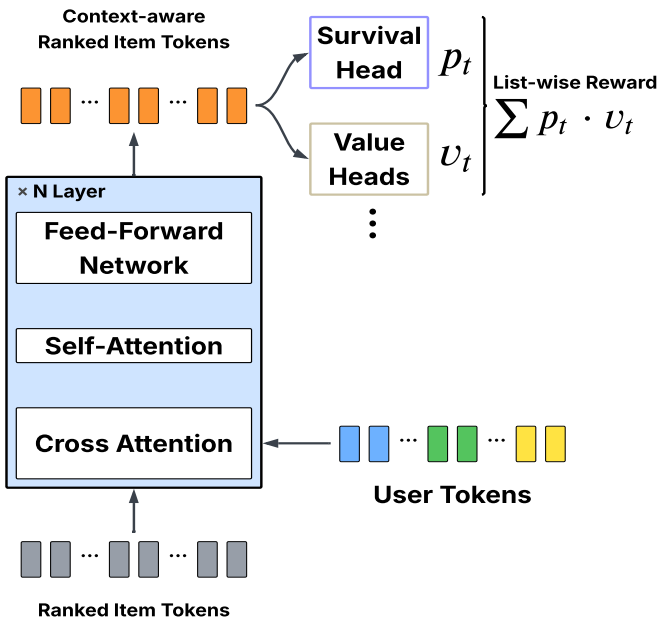

Figure 2: Architecture of the reward model. A transformer backbone encodes ranked item tokens via cross-attention to user tokens and self-attention over the ranked sequence, producing context-aware tokens that feed a survival head and K value heads.

3.5 Generator

The generator constructs ordered lists by autoregressively selecting items from the upstream candidate set. This candidate-aware design substantially reduces the search space while preserving sequential dependencies. The reward model provides training signals that guide the generator toward producing high-quality lists.

3.5.1 Generator Model Architecture. The generator follows an encoder–decoder architecture, as illustrated in Figure 3. Given a user u and a candidate set $X = \{I_1, \ldots, I_m\}$, the encoder first transforms candidate item tokens into contextualized candidate representations. This encoder is largely the same as that used in the reward model, except that here it operates on candidate item tokens rather than ranked item tokens. In this way, the encoder summarizes the candidate pool under the user condition and provides a dynamic, context-dependent item vocabulary for generation.

The decoder generates the ranked list autoregressively. At generation step t, it models the conditional distribution $P_\theta(I_{l_t} \mid u, X, I_{l_{<t}})$ over the candidate set. Each decoder layer consists of cross-attention to the encoded candidate representations and user tokens, causal self-attention over previously selected items, and a feed-forward network. Cross-attention injects user-specific and candidate-aware context into the decoding process, while causal self-attention captures dependencies among previously selected items and enforces sequential generation.

A key component of the generator is the matching-based selection mechanism. Instead of predicting over the full item vocabulary, the decoder produces a step-specific query representation q_t at each position, which serves as a query for selecting the next item from the candidate set. This representation is then matched against the contextualized candidate item tokens $\{z_1, \ldots, z_m\}$ generated by the encoder.

Specifically, for each candidate item I_j that has not yet been selected, we compute a matching score:

$$s_t(j) = q_t^\top z_j, \quad I_j \in X_{\text{rem}}^{(t)}, \tag{12}$$

where $X_{\text{rem}}^{(t)}$ denotes the remaining candidate set at step t. The probability of selecting the next item is given by:

$$P_\theta(I_j \mid u, X, I_{l_{<t}}) = \frac{\exp(s_t(j))}{\sum_{I_{j'} \in X_{\text{rem}}^{(t)}} \exp(s_t(j'))}. \tag{13}$$

The next item is selected according to this distribution and removed from the candidate set via masking, ensuring that it will not be selected again. This process is repeated until the full ranked list is generated.

This matching-based decoding strategy enables efficient generation by restricting the action space to the candidate set, while still capturing sequential dependencies and listwise interactions through a dynamic, context-dependent item vocabulary induced by the encoded candidate representations.

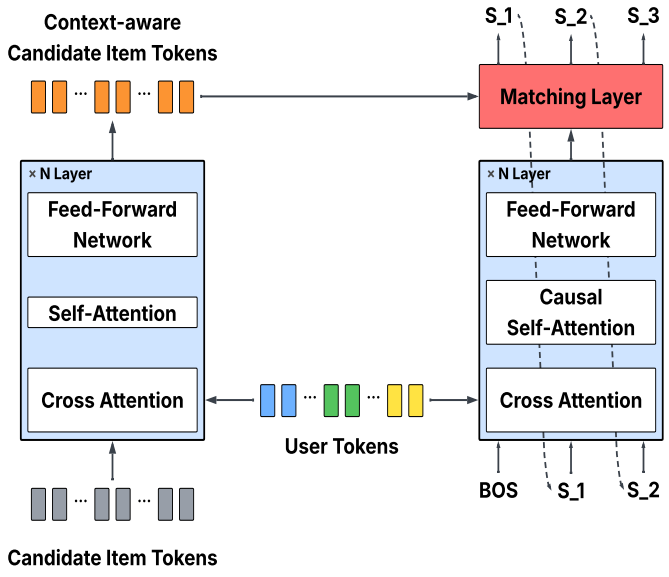

Figure 3: Architecture of the generator. Candidate items are encoded into context-aware item tokens, and a decoder autoregressively generates the re-ranked list. At each step, a matching layer selects the next item from the candidate set based on the decoder's query, with masking to avoid reselection.

3.5.2 Pre-training on Logged Exposure Data. We first pre-train the generator via supervised learning to provide a stable relevance prior before reward-guided post-training. Specifically, we use logged exposure data from the recommendation system, where each training instance consists of a user u, a candidate set X, and a historical exposed ordered list

$$L = (I_{l_1}, I_{l_2}, \ldots, I_{l_n}).$$

The pre-training objective is to maximize the likelihood of the observed list under the autoregressive generator:

$$\mathcal{L}_{\text{pre}} = -\log P_\theta(L \mid u, X) = -\sum_{t=1}^{n} \log P_\theta(I_{l_t} \mid u, X, I_{l_{<t}}), \tag{14}$$

where θ denotes the generator parameters.

This stage trains the generator to imitate historically exposed ordered lists under the candidate-aware generation setting, enabling it to learn a strong relevance prior and the sequential structure of recommendation lists from logged data. The resulting pre-trained generator provides a strong initialization for the subsequent reward-guided post-training stage.

3.5.3 Post-training via Reward-Guided Preference Alignment. After pre-training, we further refine the generator through reward-guided preference alignment. The key idea is to construct a diverse pool of candidate lists for each training instance, score them with the pretrained reward model, and then optimize the generator to align with the reward-induced listwise supervision.

List Pool Construction. For each training instance with user u and candidate set X, we construct a list pool

$$\mathcal{P}(u, X) = \{L^{(1)}, L^{(2)}, \ldots, L^{(B)}\},$$

where each $L^{(b)}$ is an ordered list over X and $B = |\mathcal{P}(u, X)|$ denotes the pool size. To ensure sufficient diversity, the pool is constructed from multiple sources, including historical exposed ordered lists, sampled lists from the current generator under different decoding strategies, and lists produced by alternative re-ranking strategies. This design mitigates self-confirmation bias and exposes the generator to a broader set of plausible ranking behaviors.

Reward-Based Supervision. Each list $L^{(b)} \in \mathcal{P}(u, X)$ in the pool is evaluated by the pretrained reward model to obtain a score $r_\phi(u, L^{(b)})$. From these scores, we derive two complementary supervision signals: a reward-induced distribution over the pool and a reward-induced ordering over the pool.

Distribution Alignment. We first convert reward scores into a soft target distribution by standardizing them within the pool. Let

$$\tilde{r}_\phi(u, L^{(b)}) = \frac{r_\phi(u, L^{(b)}) - \bar{r}_{\mathcal{P}}}{\sigma_{\mathcal{P}} + \epsilon}, \tag{15}$$

where $\bar{r}_{\mathcal{P}} = \frac{1}{|\mathcal{P}(u,X)|} \sum_{L' \in \mathcal{P}(u,X)} r_\phi(u, L')$ and $\sigma_{\mathcal{P}}$ are the mean and standard deviation of $r_\phi(u, \cdot)$ over $\mathcal{P}(u, X)$, and $\epsilon > 0$ is a small constant for numerical stability. The reward-induced reference distribution is then the softmax of these standardized rewards:

$$\pi_\phi(L^{(b)} \mid u, X) = \frac{\exp\big(\tilde{r}_\phi(u, L^{(b)})\big)}{\sum_{L' \in \mathcal{P}(u,X)} \exp\big(\tilde{r}_\phi(u, L')\big)}. \tag{16}$$

This construction follows the group-relative reference policy of GoalRank [26]: within-pool standardization makes π_ϕ robust to bounded biases in the reward model, since the induced ordering is preserved whenever reward gaps within $\mathcal{P}(u, X)$ are large relative to the bias. We adopt this principle as one of two complementary supervision signals, and further reuse the standardized rewards to modulate the ranking-alignment temperatures below.

To align the generator with this target, we define the generator-induced distribution over the same list pool:

$$\pi_\theta(L^{(b)} \mid u, X) = \frac{\exp\Big(\log P_\theta(L^{(b)} \mid u, X)\Big)}{\sum_{L' \in \mathcal{P}(u,X)} \exp(\log P_\theta(L' \mid u, X))}. \tag{17}$$

We then optimize a cross-entropy loss:

$$\mathcal{L}_{\text{dist}} = -\sum_{L \in \mathcal{P}(u,X)} \pi_\phi(L \mid u, X) \log \pi_\theta(L \mid u, X). \tag{18}$$

Ranking Alignment. Besides the distribution signal, we also enforce consistency between the ordering induced by the reward model and that induced by the generator over the same list pool. Following the Preference Ranking Optimization (PRO) framework [19], we extend pair-wise contrast to an iterative one-vs-rest contrast over the entire ranking, with contrast temperatures modulated by the standardized reward gaps.

To keep the contrast strength comparable to $\mathcal{L}_{\text{dist}}$ and independent of the absolute scale of r_ϕ, we measure reward gaps in the same standardized space defined in Eq. (15). Without loss of generality, assume the lists in the pool are sorted in descending order of reward, so that $\tilde{r}_\phi(u, L^{(1)}) \geq \tilde{r}_\phi(u, L^{(2)}) \geq \cdots \geq \tilde{r}_\phi(u, L^{(B)})$. At step k, we treat $L^{(k)}$ as the positive sample and $\{L^{(k')}\}_{k'>k}$ as negatives, contrasting them through a softmax over the generator log-likelihoods scaled by the corresponding reward gaps:

$$\mathcal{L}_{\text{rank}}^{(k)} = -\log \frac{\exp\big(\log P_\theta(L^{(k)} \mid u, X) / \mathcal{T}_{k,k}\big)}{\sum_{k'=k}^{B} \exp\big(\log P_\theta(L^{(k')} \mid u, X) / \mathcal{T}_{k,k'}\big)}, \tag{19}$$

where the per-pair temperature $\mathcal{T}_{k,k'}$ for $k' > k$ is determined by the standardized reward gap between $L^{(k)}$ and $L^{(k')}$:

$$\mathcal{T}_{k,k'} = \frac{1}{\max\big(\tilde{r}_\phi(u, L^{(k)}) - \tilde{r}_\phi(u, L^{(k')}),\ \epsilon\big)}, \qquad k' > k, \tag{20}$$

with $\epsilon > 0$ a small constant that prevents division by zero when two lists have numerically indistinguishable standardized rewards. The positive sample's own temperature is set to the smallest among its paired temperatures, $\mathcal{T}_{k,k} = \min_{k'>k} \mathcal{T}_{k,k'}$, so that its log-likelihood is scaled with the sharpest factor within the step k. Larger standardized reward gaps therefore yield smaller temperatures and sharper contrasts, allocating more supervision capacity to pairs the reward model can confidently distinguish; measuring gaps in the standardized space $\tilde{r}_\phi$ additionally makes the temperature scale-free, adapting to within-pool reward dispersion rather than the absolute magnitude of r_ϕ.

The overall ranking alignment loss aggregates contrasts across all positions in the ranking:

$$\mathcal{L}_{\text{rank}} = \sum_{k=1}^{B-1} \mathcal{L}_{\text{rank}}^{(k)}. \tag{21}$$

By iteratively promoting the current top candidate against all lower-ranked ones, this objective progressively aligns the generator's

likelihood ranking over $\mathcal{P}(u, X)$ with the ranking induced by the reward model.

Overall Objective. The final post-training objective is:

$$\mathcal{L}_{\text{post}} = \mathcal{L}_{\text{dist}} + \lambda_{\text{rank}} \mathcal{L}_{\text{rank}}, \tag{22}$$

where λ_{rank} controls the trade-off between distribution alignment and ranking consistency. This post-training stage refines the generator to better align with the reward-induced listwise preferences over the candidate pool.

3.6 Training and Inference

3.6.1 Reward Model Training. We train the reward model on logged recommendation data with observed exposure and engagement signals. The exposure component is learned from user browsing trajectories, while the value component is learned from item-level engagement signals on exposed positions. These two components are jointly optimized under the unified formulation to obtain a listwise reward model $r_\phi(u, L)$.

3.6.2 Generator Training. The generator is trained in two stages. In pre-training, we apply supervised learning on logged exposure lists, where the generator learns to model the conditional distribution $P_\theta(L \mid u, X)$ via autoregressive next-item prediction.

In post-training, we refine the generator through reward-guided preference alignment. For each (u, X), we construct a list pool, score lists using the reward model, and optimize the generator to align with reward-induced preferences derived from the reward model.

3.6.3 Generator Inference. At serving time, only the generator is used; the reward model is needed only during training. Given user and candidate representations, the generator produces an ordered list autoregressively over the candidate set. At each step, the decoder's query representation q_t is matched against the remaining candidate item tokens to compute the selection distribution, and the next item is selected greedily by taking the argmax, followed by masking to prevent reselection. We adopt greedy decoding for low-latency serving; beam search or stochastic sampling can be used when richer exploration is desired. This candidate-aware decoding enables efficient re-ranking without operating over the full item space.

4 EXPERIMENTS

We evaluate ReAlign through two complementary studies: an offline comparison against strong baselines on two public benchmarks, and ablation studies validating each design choice.

4.1 Experimental Settings

4.1.1 Datasets. We evaluate on two public datasets with complementary characteristics.

- **KuaiRand**[1] is a large-scale randomly logged dataset from Kuaishou with multi-objective engagement signals (watch time, likes, follows, comments). Random item insertion yields unbiased exposure signals. The exit position is inferred as the first position after which all subsequent play times fall below a minimum threshold.
- **Avito**[2] contains user search page logs with binary ad click signals. The exit position is inferred at the last clicked ad, with subsequent ads treated as unobserved.

4.1.2 Evaluation Metrics. We adopt two ranking metrics:

- **NDCG** (Normalized Discounted Cumulative Gain) measures ranked list quality with a logarithmic position discount that rewards relevant items at higher positions.
- **uAUC** (User-level Generalized AUC) is the generalized AUC computed per user and averaged across users. Its generalized form supports non-binary engagement labels (e.g., watch time), and the per-user aggregation is more robust to user activity imbalance than global AUC.

4.1.3 Baselines. We compare against three categories: **pointwise models** that score items independently (DNN [5]); **listwise contextual models** that capture mutual influence within a list (PRM [15]); and **permutation-level models** that evaluate ordered lists as a whole (PIER [18], CAVE [27]).

4.1.4 Implementation Details. All models are implemented in PyTorch. The user encoder extracts 8 interest tokens from the long-term sequence and uses the most recent $N = 20$ interactions as short-term tokens. The item encoder uses a 64-dim ID embedding combined with side information projected to the same dimension via element-wise addition. The transformer backbone of both the reward model and generator has 3 layers, 4 attention heads, and hidden dimension 128, with learned position embeddings matching the item token dimension. We set $K = 1$ on Avito (click) and $K = 4$ on KuaiRand (watch time, like, follow, comment), with uniform weights $\{w_k\}$ and log-uncertainty parameters $\{s_k\}$ initialized to 0. Models are optimized with AdamW (learning rate 1×10^{-4}, batch size 256), with early stopping on validation NDCG (patience 3).

For all evaluator-based approaches, we follow the MG-E protocol [22]: a fixed set of candidate permutations is scored by the reward model, and the highest-scoring permutation is selected.

4.2 Overall Performance

Table 1 summarizes the offline performance on KuaiRand and Avito. For fair comparison with scoring-based re-rankers, we report two variants: **Evaluator Only** uses our reward model as a list scorer over candidate permutations, matching the inference protocol of the baselines; **ReAlign** is the full framework where the generator produces the final list after reward-guided preference alignment.

Two observations emerge. First, Evaluator Only already surpasses all baselines on both datasets and metrics, which we attribute to the survival-based exposure modeling and the disentangled multi-objective value estimation. Second, ReAlign further improves over Evaluator Only, showing that reward-guided preference alignment effectively transfers the reward model's signal to the generator.

4.3 Ablation Study

We conduct three sets of ablations to validate key design choices: (i) the reward model architecture, (ii) the post-training objective components, and (iii) the two-stage generator training pipeline.

[1] https://kuairand.com/

[2] https://www.kaggle.com/c/avito-context-ad-clicks

Table 1: Overall performance comparison on KuaiRand and Avito datasets.

	KuaiRand		Avito	
	NDCG	uAUC	NDCG	uAUC
DNN	0.7965	0.6781	0.7022	0.6956
PRM	0.8180	0.6973	0.7182	0.7019
PIER	0.8312	0.7091	0.7281	0.7183
CAVE	0.8450	0.7196	0.7372	0.7258
Evaluator Only	0.8523	0.7236	0.7450	0.7326
ReAlign	**0.8551**	**0.7258**	**0.7476**	**0.7342**

All experiments are conducted on KuaiRand, whose richer multi-objective signals make it more representative for examining fine-grained design choices.

4.3.1 Reward Model Ablation. We construct two variants targeting the core modeling components:

- **w/o Survival** replaces the survival-based exposure module with a uniform reaching probability, removing position-dependent exposure discounting.
- **w/o Multi-Obj** collapses all engagement objectives into a single prediction head, removing both the per-objective heads and the uncertainty-based weighting.

Table 2 shows that both variants degrade performance. Ablating the survival module yields the larger drop, confirming that position-dependent exposure discounting captures genuine user browsing behavior and is critical for accurate listwise reward estimation. Ablating the multi-objective design causes a smaller but consistent drop, showing that disentangling heterogeneous engagement signals further improves estimation beyond a single-objective formulation.

Table 2: Ablation study of the reward model components.

	NDCG	uAUC
Full Reward Model	**0.8551**	**0.7258**
w/o Survival	0.8341	0.7081
w/o Multi-Obj	0.8487	0.7197

4.3.2 Post-training Loss Ablation. We compare the full loss against three variants:

- **Only** $\mathcal{L}_{\text{dist}}$ drops the ranking alignment term, supervising the generator only through the reward-induced distribution.
- **Only** $\mathcal{L}_{\text{rank}}$ drops the distribution alignment term, supervising the generator only through the PRO-style one-to-N contrast.
- **w/o Standardization** replaces the pool-standardized reward $\tilde{r}_\phi$ in Eq. (15) with the raw reward r_ϕ in both $\mathcal{L}_{\text{dist}}$ and the temperature of $\mathcal{L}_{\text{rank}}$.

Table 3 shows that all three variants degrade performance. Removing within-pool standardization yields the largest drop, confirming that a pool-adaptive scale is critical: standardization keeps contrast strength comparable across the two terms and prevents either from being dominated by the absolute scale of r_ϕ. Between the two loss components, dropping $\mathcal{L}_{\text{dist}}$ degrades more than dropping $\mathcal{L}_{\text{rank}}$, indicating that the reward-induced distribution carries the primary alignment signal, while the ranking term provides complementary sharpening of relative order. The two terms are nonetheless complementary, as the full objective outperforms both single-loss variants.

Table 3: Ablation study of the post-training loss components.

	NDCG	uAUC
Full Post-training	**0.8551**	**0.7258**
Only $\mathcal{L}_{\text{dist}}$	0.8544	0.7247
Only $\mathcal{L}_{\text{rank}}$	0.8530	0.7242
w/o Standardization	0.8527	0.7233

4.3.3 Training Stage Ablation. We compare the full two-stage pipeline against two single-stage variants:

- **Pre-training only** skips post-training and uses the pre-trained generator as the final model.
- **Post-training only** skips pre-training and applies reward-guided preference alignment from random initialization, with no relevance prior in place.

Table 4 shows that both single-stage variants underperform the full pipeline. Post-training alone reaches competitive performance and slightly outperforms pre-training only, but the full pipeline outperforms both, indicating that pre-training establishes a stable starting policy and post-training refines it to align with the reward-induced listwise preferences.

Table 4: Ablation study of the training stage.

	NDCG	uAUC
Pre-training + Post-training	**0.8551**	**0.7258**
Pre-training only	0.8305	0.7078
Post-training only	0.8369	0.7131

5 CONCLUSION

We proposed **ReAlign**, a reward-guided preference alignment framework for generative re-ranking. ReAlign couples a decomposable listwise reward model, based on survival exposure and multi-objective value estimation, with a candidate-aware autoregressive generator trained through supervised pre-training and reward-guided post-training. Extensive offline experiments on KuaiRand and Avito show that ReAlign consistently outperforms strong baselines, and ablation studies confirm the contribution of each design choice. Future work includes extending the reward model to delayed, long-horizon objectives and applying online policy distillation with token-level supervision to compress the generator into a lightweight student model for low-latency serving.

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] 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).

[3] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 2942–2951.

[4] Ben Chen, Xian Guo, Siyuan Wang, Zihan Liang, Yue Lv, Yufei Ma, Xinlong Xiao, Bowen Xue, Xuxin Zhang, Ying Yang, et al. 2025. Onesearch: A preliminary exploration of the unified end-to-end generative framework for e-commerce search. *arXiv preprint arXiv:2509.03236* (2025).

[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] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965* (2025).

[7] Yufei Feng, Yu Gong, Fei Sun, Junfeng Ge, and Wenwu Ou. 2021. Revisit recommender system in the permutation prospective. *arXiv preprint arXiv:2102.12057* (2021).

[8] Yufei Feng, Binbin Hu, Yu Gong, Fei Sun, Qingwen Liu, and Wenwu Ou. 2021. GRN: Generative Rerank Network for Context-wise Recommendation. *arXiv preprint arXiv:2104.00860* (2021).

[9] Ruidong Han, Bin Yin, Shangyu Chen, He Jiang, Fei Jiang, Xiang Li, Chi Ma, Mincong Huang, Xiaoguang Li, Chunzhen Jing, et al. 2025. Mtgr: Industrial-scale generative recommendation framework in meituan. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 5731–5738.

[10] Clark Mingxuan Ju, Liam Collins, Leonardo Neves, Bhuvesh Kumar, Louis Yufeng Wang, Tong Zhao, and Neil Shah. 2025. Generative Recommendation with Semantic IDs: A Practitioner's Handbook. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 6420–6425.

[11] Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7482–7491.

[12] Zhijie Lin, Zhuofeng Li, Chenglei Dai, Wentian Bao, Shuai Lin, Enyun Yu, Haoxiang Zhang, and Liang Zhao. 2025. GReF: A Unified Generative Framework for Efficient Reranking via Ordered Multi-token Prediction. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 5879–5887.

[13] Tianqi Liu, Zhen Qin, Junru Wu, Jiaming Shen, Misha Khalman, Rishabh Joshi, Yao Zhao, Mohammad Saleh, Simon Baumgartner, Jialu Liu, et al. 2025. Lipo: Listwise preference optimization through learning-to-rank. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. 2404–2420.

[14] 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.

[15] 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.

[16] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.

[17] Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. 2024. Non-autoregressive generative models for reranking recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 5625–5634.

[18] Xiaowen Shi, Fan Yang, Ze Wang, Xiaoxu Wu, Muzhi Guan, Guogang Liao, Wang Yongkang, Xingxing Wang, and Dong Wang. 2023. Pier: Permutation-level interest-based end-to-end re-ranking framework in e-commerce. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4823–4831.

[19] Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18990–18998.

[20] Shuli Wang, Xue Wei, Senjie Kou, Chi Wang, Wenshuai Chen, Qi Tang, Yinhua Zhu, Xiong Xiao, and Xingxing Wang. 2025. Nlgr: Utilizing neighbor lists for generative rerank in personalized recommendation systems. In *Companion Proceedings of the ACM on Web Conference 2025*. 530–537.

[21] Ben Xue, Dan Liu, Lixiang Wang, Mingjie Sun, Peng Wang, Pengfei Zhang, Shaoyun Shi, Tianyu Xu, Yunhao Sha, Zhiqiang Liu, et al. 2026. Generative Recommendation for Large-Scale Advertising. *arXiv preprint arXiv:2602.22732* (2026).

[22] Hailan Yang, Zhenyu Qi, Shuchang Liu, Xiaoyu Yang, Xiaobei Wang, Xiang Li, Lantao Hu, Han Li, and Kun Gai. 2025. Comprehensive list generation for multi-generator reranking. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2298–2308.

[23] Yuhao Yang, Zhi Ji, Zhaopeng Li, Yi Li, Zhonglin Mo, Yue Ding, Kai Chen, Zijian Zhang, Jie Li, Shuanglong Li, et al. 2025. Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations. *arXiv preprint arXiv:2503.02453* (2025).

[24] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Michael He, et al. 2024. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *arXiv preprint arXiv:2402.17152* (2024).

[25] Jun Zhang, Yi Li, Yue Liu, Changping Wang, Yuan Wang, Yuling Xiong, Xun Liu, Haiyang Wu, Qian Li, Enming Zhang, et al. 2025. GPR: Towards a Generative Pre-trained One-Model Paradigm for Large-Scale Advertising Recommendation. *arXiv preprint arXiv:2511.10138* (2025).

[26] Kaike Zhang, Xiaobei Wang, Shuchang Liu, Hailan Yang, Xiang Li, Lantao Hu, Han Li, Qi Cao, Fei Sun, and Kun Gai. 2025. Goalrank: Group-relative optimization for a large ranking model. *arXiv preprint arXiv:2509.22046* (2025).

[27] Kaike Zhang, Xiaobei Wang, Xiaoyu Yang, Shuchang Liu, Hailan Yang, Xiang Li, Fei Sun, and Qi Cao. 2025. From generation to consumption: Personalized list value estimation for re-ranking. *arXiv preprint arXiv:2508.02242* (2025).

[28] Guorui Zhou, Hengrui Hu, Hongtao Cheng, Huanjie Wang, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Lu Ren, Liao Yu, et al. 2025. Onerec-v2 technical report. *arXiv preprint arXiv:2508.20900* (2025).

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