LIFT: A LIFECYCLE-AWARE INTERACTION FACTORIZATION TRANSFORMER FOR UNIFIED RETRIEVAL AND RANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cascaded recommender systems use the same user history for retrieval and ranking, while the two stages have access to different information at different points of an interaction. We propose the Lifecycle-aware Interaction Factorization Transformer (LIFT), which decomposes each interaction into ordered Request, Item, Context, and Action states and models them as a causal sequence. Retrieval reads the Request state, while ranking reads the Context state, allowing both tasks to share history modeling while preserving stage-specific information. LIFT instantiates this representation with Role-Conditioned Attention and a lightweight Pre-LN Bias. On ML-20M and Taobao, LIFT achieves the highest Joint Score among the evaluated joint models, improving over the strongest baselines by 4.9% and 3.6%, respectively. Loss-weight sweeps show favorable retrieval–ranking trade-offs, while ablations and scaling analyses further examine lifecycle sequence construction, model components, and capacity settings.

1 INTRODUCTION

Modern recommender systems commonly use a retrieval–ranking cascade (Qin et al., 2022; Liu et al., 2022). Retrieval selects candidates from a large item collection (Gao et al., 2020; Zhu et al., 2018), while ranking predicts user feedback (Zhou et al., 2019; Lian et al., 2018) using the retrieved candidates and additional contextual features. Both stages depend on the same user history, creating substantial opportunities for shared historical modeling (Gao et al., 2020). Recent unified architectures reuse history representations across tasks and recommendation stages through shared modeling, joint training, and cross-stage caching (Deng et al., 2025; Dai et al., 2026; Li et al., 2026), yet retrieval and ranking operate with different information available within an interaction.

Sharing history computation alone does not resolve the representational difference between retrieval and ranking. Retrieval makes a decision from the user history and the current request, whereas ranking operates after a candidate item has been introduced and can additionally use candidate-side context. Sequential models, however, commonly represent each interaction as an item- or event-level state (Hidasi et al., 2015; Kang & McAuley, 2018; Zhai et al., 2024). This leaves the information-availability boundaries within an interaction implicit. A unified retrieval–ranking model therefore benefits from representing the intermediate states through which an interaction evolves.

Figure 1 illustrates the interaction lifecycle considered in this work. A Request is linked to an Item through retrieval; contextual information is then incorporated, and ranking predicts the resulting user Action. The arrows indicate the logical order in which information becomes available during an interaction, while the four roles together form a single interaction record.

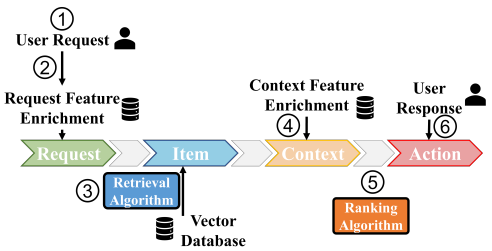


Figure 1: Lifecycle of a recommendation interaction.

Following this progression, we propose the Lifecycle-aware Interaction Factorization Transformer (LIFT). LIFT factorizes each interaction into four ordered states—Request, Item, Context, and Action—according to when information be-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

comes available, and serializes them into a causal sequence. Retrieval reads the Request state, while ranking reads the Context state after the candidate and its contextual features have become available.

The serialized lifecycle sequence also supports shared historical computation across recommendation stages. During online serving, the historical sequence is prefilled once and its cached states are reused by retrieval and ranking, which read the Request and Context states, respectively.

We instantiate this representation with Role-Conditioned Attention, which shares Query and Key projections while conditioning Value and Output transformations on lifecycle roles. A lightweight Pre-LN Bias further conditions the feed-forward path on the current role. Retrieval and ranking retain their task-specific candidate processing, supervision, and prediction heads while sharing the causal history encoder.

We compare LIFT with single-task models, multi-task models, and unified sequence models on ML-20M and Taobao. LIFT achieves the highest Joint Score among the evaluated joint models on both datasets. On ML-20M, it obtains a score of 0.2724, compared with 0.2597 for the strongest baseline, HSTU, a relative improvement of 4.9%. On Taobao, it improves over the strongest baseline by 3.6%. Loss-weight sweeps show a more favorable retrieval–ranking quality trade-off for LIFT over the evaluated range. Ablation studies further examine lifecycle sequence construction, Role-Conditioned Attention, and Pre-LN Bias.

Our contributions are summarized as follows:

- We propose lifecycle-aware interaction factorization, representing each interaction as ordered Request, Item, Context, and Action states and assigning stage-aligned readouts to retrieval and ranking.
- We instantiate lifecycle-aware interaction factorization with a causal history encoder that uses Role-Conditioned Attention and a lightweight additive Pre-LN Bias, while preserving the task-specific interfaces of retrieval and ranking.
- We conduct controlled experiments on ML-20M and Taobao, evaluating joint recommendation quality, retrieval–ranking trade-offs, and the effects of lifecycle design choices and model components.

2 Related Work

2.1 Sequential Recommendation Modeling

Sequential recommendation organizes user behavior as a temporally ordered interaction sequence to model behavioral order, evolving interests, and long-range dependencies (Quadrana et al., 2018; Pan et al., 2026; Zhou et al., 2019). GRU4Rec encodes behavior sequences with recurrent neural networks, while SASRec introduces self-attention for sequential recommendation (Hidasi et al., 2015; Kang & McAuley, 2018). Subsequent work extends Transformer-based sequence modeling through bidirectional representations, temporal intervals, and self-supervised objectives (Sun et al., 2019; Li et al., 2020; Zhou et al., 2020). HSTU further formulates recommendation as sequence transduction for large-scale, non-stationary behavior streams (Zhai et al., 2024). LIFT represents the request as a distinct token, enabling the model to make explicit use of request-side information that is available at retrieval time.

2.2 Unified Modeling of Retrieval and Ranking

Prior work has explored different levels of integration between retrieval and ranking (Qin et al., 2022). CoRR jointly improves a retriever and a ranker through adaptive negative sampling and knowledge distillation (Huang et al., 2023). More recent work moves toward unified model architectures. UniGRF reformulates retrieval and ranking as generative objectives within a common framework (Zhang et al., 2025). OneRec replaces the cascaded pipeline with a generative recommender and iterative preference alignment (Deng et al., 2025). OnePiece uses structured context for both retrieval and ranking, together with block-wise latent reasoning and progressive multi-task training (Dai et al., 2026). UniPinRec unifies the input format, model architecture, and training pipeline, and reuses user-history computation across stages through cross-stage KV caching (Li et al., 2026). LIFT

explores a different representation granularity by factorizing each interaction according to information availability. Existing sequential formulations typically associate an interaction or action event with a single sequence state, whereas LIFT exposes multiple causal states within the interaction itself. Their order follows when Request, Item, Context, and Action information becomes available, allowing retrieval and ranking to read distinct lifecycle states within the same causal history model.

2.3 Multi-Task Learning and Structured Parameter Sharing

Multi-task learning coordinates parameter use across multiple prediction tasks through shared representations, gated experts, or a combination of shared and task-specific experts (Ruder, 2017). Shared-bottom parameter sharing follows the classical multi-task learning paradigm (Caruana, 1997). MMoE uses task-specific gating over shared experts (Ma et al., 2018), while PLE further organizes shared and task-specific experts (Tang et al., 2020). LIFT combines shared history encoding with stage-aligned state readouts: retrieval and ranking reuse the same causal interaction sequence while reading different states suited to their respective stages.

3 Method

3.1 Problem Setup and System Overview

In LIFT, the retrieval task generates candidate items based on the user history and the current request, while the ranking task predicts user feedback for candidate items. During training, both tasks share the same lifecycle sequence and historical encoding results, while differing in state readout positions, candidate processing, negative sampling, and supervision. During online serving, the ranking stage further introduces candidate-related features after the retrieval candidates have been determined. LIFT unifies the two tasks within a common lifecycle-based sequence modeling framework.

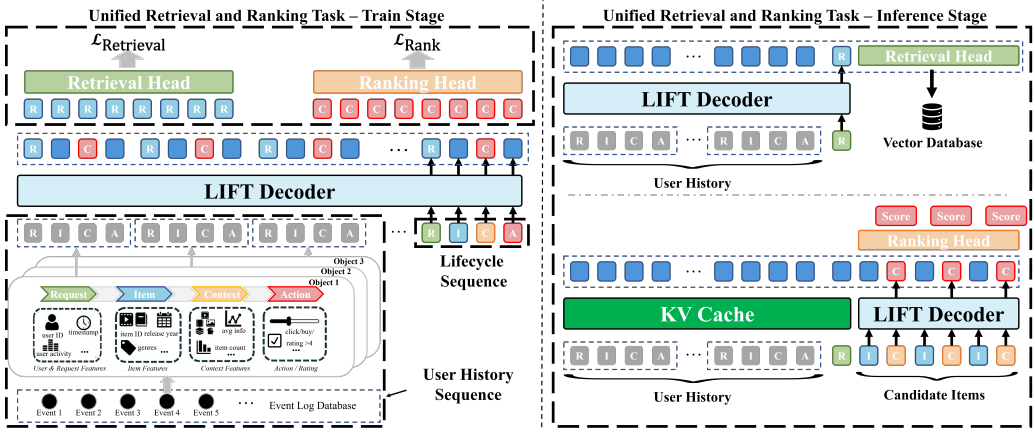


Figure 2: Overview of LIFT's unified training and online serving pipelines.

Figure 2 summarizes the overall architecture and information flow of LIFT, including the shared lifecycle sequence and the LIFT Decoder, the Request- and Context-based readout branches, and the online retrieval–ranking cascade.

LIFT organizes user interactions according to four lifecycle roles: Request, Item, Context, and Action. For a user u with an interaction sequence of length T, let Π_{Req} and Π_{Ctx} denote the readout operators for the Request and Context states, respectively. The unified encoding and task prediction process is given by:

$$\begin{aligned} \mathbf{Z}_{u,\le T} &= \left[\mathbf{z}^R_{u,1}, \mathbf{z}^I_{u,1}, \mathbf{z}^C_{u,1}, \mathbf{z}^A_{u,1}, \ldots, \mathbf{z}^R_{u,T}, \mathbf{z}^I_{u,T}, \mathbf{z}^C_{u,T}, \mathbf{z}^A_{u,T}\right], \\ \mathbf{H}_{u,\le T} &= F_\theta\left(\mathbf{Z}_{u,\le T}\right), \end{aligned} \tag{1}$$

$$\hat{\mathbf{y}}^{\text{ret}}_{u,\le T} = g_{\text{ret}}\left(\Pi_{\text{Req}}\left(\mathbf{H}_{u,\le T}\right)\right), \quad \hat{\mathbf{y}}^{\text{rank}}_{u,\le T} = g_{\text{rank}}\left(\Pi_{\text{Ctx}}\left(\mathbf{H}_{u,\le T}\right)\right). \tag{2}$$

Here, $\mathbf{z}^R$, $\mathbf{z}^I$, $\mathbf{z}^C$, and $\mathbf{z}^A$ denote Request, Item, Context, and Action token representations, respectively. The lifecycle-aware representation is described in Section 3.2.

In Eq. (1), F_θ denotes the shared causal Transformer Decoder (Vaswani et al., 2017), and $\mathbf{H}_{u,\le T}$ denotes the hidden-state sequence aligned with the lifecycle tokens. The outputs $\hat{\mathbf{y}}^{\text{ret}}_{u,\le T}$ and $\hat{\mathbf{y}}^{\text{rank}}_{u,\le T}$

denote the vectorized candidate-level outputs of the retrieval and ranking tasks at valid temporal positions, respectively. The function g_{ret} maps Request states to outputs over the retrieval candidate space, while g_{rank} maps Context states to outputs over the ranking candidate set. The Decoder uses shared parameters to model dependencies across historical interactions while retaining role-dependent local state transformations. The detailed interaction modeling mechanism is presented in Section 3.3.

During training, the two tasks produce supervision through separate task heads and loss functions. The shared LIFT Decoder is jointly optimized by both tasks, while candidate construction, negative sampling, and supervision labels remain task-specific:

$$\begin{gathered}\mathcal{L}_{\mathrm{ret}} = \ell_{\mathrm{ret}}\left(\widehat{\mathbf{y}}^{\mathrm{ret}}_{u,\le T}, \mathbf{y}^{\mathrm{ret}}_{u,\le T}, \mathcal{N}^{\mathrm{ret}}\right), \quad \mathcal{L}_{\mathrm{rank}} = \ell_{\mathrm{rank}}\left(\widehat{\mathbf{y}}^{\mathrm{rank}}_{u,\le T}, \mathbf{y}^{\mathrm{rank}}_{u,\le T}, \mathcal{N}^{\mathrm{rank}}\right). \\ \mathcal{L} = \lambda_{\mathrm{ret}}\mathcal{L}_{\mathrm{ret}} + \lambda_{\mathrm{rank}}\mathcal{L}_{\mathrm{rank}},\end{gathered} \tag{3}$$

Here, $\mathbf{y}^{\mathrm{ret}}_{u,\le T}$ denotes the target-item labels, while $\mathbf{y}^{\mathrm{rank}}_{u,\le T}$ denotes the ranking labels constructed from the corresponding Action feedback. $\mathcal{N}^{\mathrm{ret}}$ and $\mathcal{N}^{\mathrm{rank}}$ represent the negative samples or candidate-sampling information used by the two tasks, respectively. The functions ℓ_{ret} and ℓ_{rank} compute and aggregate the task losses over valid temporal positions according to their respective supervision and sampling strategies.

During online serving at time t, LIFT first performs a single shared prefill over the user's historical lifecycle sequence and caches the resulting Key–Value (KV) states as $\mathrm{KV}_{u,<t}$. The retrieval stage reads the Request state, computes scores for candidate items, and generates a candidate set. The ranking stage then constructs the corresponding Item and Context inputs from the retrieved candidates and computes their ranking scores.

$$\begin{aligned}\widehat{\mathcal{C}}^{\mathrm{ret}}_{u,t} &= \mathrm{RetrieveByTopK}\left(\mathrm{KV}_{u,<t}; \widetilde{\mathbf{z}}^{R}_{u,t}; \mathcal{V}^{\mathrm{ret}}_{t}\right), \\ \widehat{\mathcal{S}}^{\mathrm{rank}}_{u,t} &= \mathrm{RankingByScore}\left(\mathrm{KV}_{u,<t}; \widetilde{\mathbf{z}}^{R}_{u,t}; \left\{(\widetilde{\mathbf{z}}^{I}_{u,t,c}, \widetilde{\mathbf{z}}^{C}_{u,t,c}) \mid c \in \widehat{\mathcal{C}}^{\mathrm{ret}}_{u,t}\right\}\right).\end{aligned} \tag{4}$$

Here, the tilde denotes representations constructed from the current request or candidate-specific features, in contrast to historical lifecycle tokens. $\mathrm{RetrieveByTopK}(\cdot)$ selects the K highest-scoring candidate items from the candidate item space $\mathcal{V}^{\mathrm{ret}}_{t}$ based on the retrieval scores, forming the input set $\widehat{\mathcal{C}}^{\mathrm{ret}}_{u,t}$ for the ranking stage. The ranking stage constructs the corresponding Item and Context tokens for this candidate set and computes ranking scores. Finally, $\mathrm{RankingByScore}(\cdot)$ reorders the candidates according to their scores to produce the ordered candidate-item sequence $\widehat{\mathcal{S}}^{\mathrm{rank}}_{u,t}$. This process allows the two tasks to share historical dependency modeling while preserving their respective candidate-processing and prediction interfaces.

3.2 Lifecycle-aware Interaction Representation

As illustrated in Figure 2, a user interaction contains information with different semantics, including a request, an item, contextual features, and user feedback. Based on the semantics of these fields and their availability during the recommendation process, we represent the interaction of user u at time t as

$$e_{u,t} = \left(x^{R}_{u,t}, x^{I}_{u,t}, x^{C}_{u,t}, x^{A}_{u,t}\right), \tag{5}$$

where the four types of fields describe the trigger condition, interaction object, candidate context, and user response, respectively. This decomposition is determined by the generation dependencies and availability of the fields, independently of how the fields are organized in a particular data table.

- $x^{R}_{u,t}$ denotes the Request information, namely the request-side inputs available before the candidate items are determined. It includes the user identifier, current query or request features, session information, temporal features, and device- or traffic-related context, which together describe the conditions that trigger the interaction (Wang et al., 2018; Zhu et al., 2018; Gao et al., 2020).
- $x^{I}_{u,t}$ denotes the Item information, namely the relatively stable attributes of the interacted item, including its identifier, category, brand, and content features. In large-scale retrieval systems, item-side representations are commonly precomputed or organized into retrieval structures for efficient candidate generation (Wang et al., 2018; Gao et al., 2020).
- $x^{C}_{u,t}$ denotes the Context information, namely the candidate-related features formed after the item is introduced into the current request and interaction scenario. Examples include time-windowed

item statistics, user–item interaction statistics, candidate sources, and scene features determined before scoring (Cheng et al., 2016; Zhou et al., 2019; Lian et al., 2018; Song et al., 2019). Statistics computed from historical behavior use only information available before the current time step. Such candidate-conditioned modeling is widely used in behavior-aware ranking models (Zhou et al., 2019).

- $x^A_{u,t}$ denotes the Action information, namely the user feedback generated after item exposure or interaction, including clicks, dwell time, collections, purchases, and ratings (Hu et al., 2008; Rendle et al., 2012). Historical interactions use the observed Action, while the ground-truth Action at the target position is excluded from the input and used only to construct the training supervision or evaluation label.

The four types of fields therefore follow this lifecycle order: $\mathrm{Request} \rightarrow \mathrm{Item} \rightarrow \mathrm{Context} \rightarrow \mathrm{Action}$. This order reflects how information becomes available during a recommendation interaction: Request specifies the decision conditions, Item identifies the interaction object, Context describes the candidate under the current scenario, and Action records the final user response. This decomposition provides multiple semantically distinct state positions within the same interaction, allowing retrieval and ranking to read historical representations corresponding to their respective prediction stages.

For any role q in the role set $\mathcal{Q} = \{R, I, C, A\}$, we further divide its input fields into a set of sparse fields $\mathcal{S}_q$ and a set of dense fields $\mathcal{D}_q$. Sparse fields are encoded using independent embedding tables, while dense fields are first normalized and then transformed by a dense-feature encoder. The two types of representations are aggregated separately, concatenated, and projected into a shared d-dimensional hidden space (Rendle, 2010; Naumov et al., 2019):

$$\begin{aligned} \mathbf{s}^q_{u,t} &= \mathrm{Agg}_{\mathrm{sparse}}\left(\left\{\mathbf{E}_{q,k}\left[x^{q,k}_{u,t}\right]\right\}_{k\in\mathcal{S}_q}\right), \\ \mathbf{d}^q_{u,t} &= \mathrm{Enc}_{\mathrm{dense},q}\left(\mathrm{Concat}_{\ell\in\mathcal{D}_q}\,\mathrm{Norm}\left(x^{q,\ell}_{u,t}\right)\right), \\ \mathbf{z}^q_{u,t} &= \mathrm{Proj}_q\left(\left[\mathbf{s}^q_{u,t}; \mathbf{d}^q_{u,t}\right]\right) \in \mathbb{R}^d, \qquad q \in \mathcal{Q}. \end{aligned} \tag{6}$$

Here, $\mathbf{E}_{q,k}$ denotes the embedding table for the k-th sparse field of role q, $\mathrm{Agg}_{\mathrm{sparse}}(\cdot)$ denotes the aggregation of sparse-field representations, $\mathrm{Norm}(\cdot)$ denotes dense-feature normalization, $\mathrm{Enc}_{\mathrm{dense},q}(\cdot)$ denotes the dense-feature encoder, and $\mathrm{Proj}_q(\cdot)$ projects the concatenated representation to the fixed hidden dimension d. Consequently, different roles can use role-specific input fields while producing lifecycle tokens in the same hidden space.

LIFT then expands each interaction into four tokens according to the lifecycle order and concatenates all historical interactions into a unified sequence:

$$\begin{aligned} \mathbf{Z}_{u,\le T} &= \mathrm{Serialize}\left(e_{u,1:T}\right) \\ &= \left[\mathbf{z}^R_{u,1}, \mathbf{z}^I_{u,1}, \mathbf{z}^C_{u,1}, \mathbf{z}^A_{u,1}, \ldots, \mathbf{z}^R_{u,T}, \mathbf{z}^I_{u,T}, \mathbf{z}^C_{u,T}, \mathbf{z}^A_{u,T}\right] \in \mathbb{R}^{4T\times d}. \end{aligned} \tag{7}$$

The lifecycle order also determines the availability of information within an interaction. Let $R \prec I \prec C \prec A$ denote the precedence relation among the four roles. Under causal modeling, the hidden state at each position can depend only on the current token and preceding tokens in the serialized sequence. Roles that occur later in the lifecycle do not contribute to its computation. Consequently, a Request state primarily reflects the historical information available when the request is formed, while a Context state can additionally use the Item and Context information of the current interaction. At a prediction position, the ground-truth Action is excluded from the input and is used only to construct the training supervision. The detailed causal-attention constraints and lifecycle-role-specific state transformations are presented in Section 3.3.

The lifecycle states provide natural prediction boundaries for the two stages. Retrieval is performed before a candidate item is selected, so its representation should depend on the historical sequence and the current Request, but not on the Item, Context, or Action of the current interaction. The Request state therefore forms the retrieval readout. Ranking is applied after a candidate item has been introduced and its candidate-dependent features are available, while the user Action remains unknown. The Context state consequently forms the ranking readout. These two states are produced by the same causal sequence model but correspond to different information sets, allowing shared historical modeling without collapsing the stage boundary between retrieval and ranking.

Through this representation, each interaction is organized as four tokens with an explicit lifecycle order. The resulting sequence preserves both cross-interaction historical dependencies and information availability within each interaction, providing retrieval and ranking with a shared input space and semantically distinct state readout positions.

3.3 Role-Conditioned Attention and Pre-LN Bias

Given the lifecycle sequence introduced in Section 3.2, let $L = 4T$ denote the number of lifecycle tokens. Let $\mathcal{Q} = \{R, I, C, A\}$ be the set of lifecycle roles and $\mathcal{Q}_{\text{ord}} = (R, I, C, A)$ their fixed order. For token position $i \in \{1, \dots, L\}$, its lifecycle role and interaction position are defined as

$$\rho(i) = (\mathcal{Q}_{\text{ord}})_{1+((i-1) \bmod 4)}, \qquad \tau(i) = \left\lfloor \frac{i-1}{4} \right\rfloor + 1. \tag{8}$$

To simplify notation, we omit the user and sequence subscripts and write $\mathbf{Z} = \mathbf{Z}_{u,\le T}$ and $\mathbf{H}^{(B)} = \mathbf{H}_{u,\le T}$. Let $b \in \{1, \dots, B\}$ index the Decoder layers, and denote the input token at position i by $\mathbf{z}_i = \mathbf{z}^{\rho(i)}_{u,\tau(i)}$.

The LIFT Decoder takes the input tokens as its zeroth-layer hidden states and produces position-aligned hidden states after B Decoder layers:

$$\begin{aligned} \mathbf{H}^{(0)} &= \mathbf{Z} = [\mathbf{z}_1, \dots, \mathbf{z}_L], \\ \mathbf{H}^{(B)} &= F_\theta(\mathbf{Z}) = \left[\mathbf{h}_1^{(B)}, \mathbf{h}_2^{(B)}, \dots, \mathbf{h}_L^{(B)}\right]. \end{aligned} \tag{9}$$

To make the parameter-sharing boundary explicit, we partition the Decoder parameters into those shared across lifecycle roles and those indexed by lifecycle role:

$$\begin{aligned} \theta_{\text{shared}} &= \left\{ \mathbf{W}_Q^{(b)}, \mathbf{W}_K^{(b)}, \theta_{\text{norm}}^{(b)}, \theta_{\text{FFN}}^{(b)} \;\middle|\; b = 1, \dots, B \right\}, \\ \theta_{\text{role}} &= \left\{ \mathbf{r}_q^{(b)}, \mathbf{W}_V^{(b,q)}, \mathbf{W}_O^{(b,q)}, \mathbf{b}_q^{(b)} \;\middle|\; b = 1, \dots, B,\ q \in \mathcal{Q} \right\}, \\ \theta &= \theta_{\text{shared}} \cup \theta_{\text{role}}. \end{aligned} \tag{10}$$

Here, $\theta_{\text{norm}}^{(b)}$ collects the normalization parameters in layer b, and $\theta_{\text{FFN}}^{(b)}$ denotes the parameters of its feed-forward sublayer. The vectors $\mathbf{r}_q^{(b)}$ and $\mathbf{b}_q^{(b)}$ are the learnable role embedding and additive bias associated with role q, respectively. Task-specific prediction heads are defined separately in Section 3.1.

The lifecycle roles differ primarily in the information they carry and the states they produce, while their dependencies are modeled in a common sequence space. We therefore share Query and Key projections for dependency matching, and condition Value and Output projections on the source and receiving roles, respectively.

At layer b, Role-Conditioned Attention models dependencies across interactions and lifecycle roles. Given the input hidden state $\mathbf{h}_i^{(b-1)}$, Query, Key, and Value are constructed through the following pre-normalization paths:

$$\begin{aligned} \widetilde{\mathbf{h}}_i^{V,(b)} &= \mathrm{LN}_{\text{attn}}^{(b)}\left(\mathbf{h}_i^{(b-1)}\right), \\ \widetilde{\mathbf{h}}_i^{QK,(b)} &= \mathrm{LN}_{\text{attn}}^{(b)}\left(\mathbf{h}_i^{(b-1)} + \mathbf{r}_{\rho(i)}^{(b)}\right), \\ \mathbf{q}_i^{(b)} &= \widetilde{\mathbf{h}}_i^{QK,(b)} \mathbf{W}_Q^{(b)}, \qquad \mathbf{k}_i^{(b)} = \widetilde{\mathbf{h}}_i^{QK,(b)} \mathbf{W}_K^{(b)}, \\ \mathbf{v}_i^{(b)} &= \widetilde{\mathbf{h}}_i^{V,(b)} \mathbf{W}_V^{(b,\rho(i))}. \end{aligned} \tag{11}$$

The role vector $\mathbf{r}_{\rho(i)}^{(b)}$ is a learnable identity vector associated with the lifecycle role at position i. Query and Key are computed from the normalized state augmented with role information, whereas Value is computed from the normalized state without the role embedding. The Value projection is selected according to the role of the source token.

Attention scores are computed using an event-position-aware Query–Key compatibility function:

$$\ell_{ij}^{(b)} = \frac{\Psi_{\text{evt}}\left(\mathbf{q}_i^{(b)}, \mathbf{k}_j^{(b)}; \tau(i), \tau(j)\right)}{\sqrt{d_h}} + M_{ij}, \qquad M_{ij} = \begin{cases} 0, & j \le i, \\ -\infty, & j > i. \end{cases} \tag{12}$$

Here, $\Psi_{\text{evt}}(\mathbf{q}_i, \mathbf{k}_j; \tau(i), \tau(j)) = (\mathbf{R}_{\tau(i)}\mathbf{q}_i)^\top (\mathbf{R}_{\tau(j)}\mathbf{k}_j)$ denotes RoPE-based Query–Key compatibility (Su et al., 2024), where $\mathbf{R}_\tau$ is the rotary transformation at interaction position τ. Thus, lifecycle tokens from the same interaction share the same rotary position, and M_{ij} is the causal mask.

Let $\boldsymbol{\ell}_{i,1:L}^{(b)} = (\ell_{i1}^{(b)}, \ldots, \ell_{iL}^{(b)})$ denote the attention-score vector from position i to the full sequence. The attention output and residual connection are defined as

$$\mathbf{u}_i^{(b)} = \mathbf{h}_i^{(b-1)} + \left[\sum_{j=1}^{L} \left[\text{softmax}\left(\boldsymbol{\ell}_{i,1:L}^{(b)}\right)\right]_j \mathbf{v}_j^{(b)}\right] \mathbf{W}_O^{(b,\rho(i))}. \tag{13}$$

The output projection $\mathbf{W}_O^{(b,\rho(i))}$ is selected according to the lifecycle role of the receiving position. Thus, the Value projection depends on the source role, whereas the Output projection depends on the receiving role. Role-Conditioned Attention models dependencies across interactions and lifecycle roles in a shared Query–Key matching space while preserving role-dependent information transformations.

Following the Pre-LN Transformer organization (Xiong et al., 2020), LIFT adds a role-specific bias on the pre-normalization path to the feed-forward sublayer. Let $\mathbf{u}_i^{(b)}$ denote the output of the attention sublayer. The FFN input and the resulting hidden state are

$$\begin{aligned} \widetilde{\mathbf{u}}_i^{(b)} &= \text{LN}_{\text{FFN}}^{(b)}\left(\mathbf{u}_i^{(b)} + \mathbf{b}_{\rho(i)}^{(b)}\right), \\ \mathbf{h}_i^{(b)} &= \mathbf{u}_i^{(b)} + \text{FFN}^{(b)}\left(\widetilde{\mathbf{u}}_i^{(b)}\right). \end{aligned} \tag{14}$$

The final Request and Context states are extracted by the task-specific readout operators defined in Section 3.1 and passed to the retrieval and ranking heads, respectively.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate LIFT on ML-20M (Harper & Konstan, 2015) and Taobao (Alibaba Tianchi, 2017). Table 1 summarizes the dataset statistics. The catalog size excludes the padding ID.

Table 1: Dataset statistics for ML-20M and Taobao.

	Dataset size		Interactions		
Dataset	Users	Items	Train	Validation	Test
ML-20M	138,493	27,278	19,153,489	562,888	283,886
Taobao	961,904	1,518,629	68,556,084	12,607,812	12,423,797

Both datasets are split chronologically. For ML-20M, training interactions span 1995-01-09 to 2013-12-31, validation covers 2014-01-01 to 2014-12-31, and test covers 2015-01-01 to 2015-03-31. For Taobao, training covers 2017-11-25 to 2017-12-01, validation is 2017-12-02, and test is 2017-12-03. We retain Taobao items with at least 5 training-period interactions and users with at least 10 retained interactions, including at least 1 training interaction (Wang et al., 2018).

Tasks and evaluation protocol. The retrieval task predicts the next interacted item and is trained with an InfoNCE objective (Oord et al., 2018) over in-batch target items, with LogQ correction (Yi et al., 2019). At evaluation, we rank items from the available catalog and filter items previously interacted with by the user. We report Recall@100 and NDCG@100. The ranking task predicts feedback for a target interaction and is trained with binary cross-entropy. For ML-20M, ratings greater than 4 are positive labels. For Taobao, purchases are positive, and the other behavior types are negative. We report AUC (Fawcett, 2006) and GAUC (Zhou et al., 2018). For joint recommendation, we define the Joint Score as

$$\text{JointScore} = \sqrt{\text{NDCG@100} \cdot \text{GAUC}}. \tag{15}$$

The geometric mean treats retrieval and ranking symmetrically and penalizes solutions that perform poorly on either stage.

Table 2: Joint Retrieval–Ranking results on ML-20M and Taobao. Joint Score is defined as $\sqrt{\text{NDCG@100} \cdot \text{GAUC}}$.

	ML-20M					Taobao				
Method	R@100	N@100	AUC	GAUC	Joint	R@100	N@100	AUC	GAUC	Joint
SharedBottom	0.2603	0.0681	0.7748	0.7058	0.2193	0.1518	0.0454	0.8950	0.8575	0.1972
MMoE	0.2702	0.0724	0.7928	0.7105	0.2268	0.1552	0.0467	0.9001	0.8588	0.2002
PLE	0.2004	0.0483	0.7649	0.6451	0.1764	0.1521	0.0455	0.8962	0.8556	0.1973
HSTU	0.3157	0.0910	0.8368	0.7410	0.2597	0.1716	0.0526	0.9099	0.8570	0.2123
UniPinRec	0.3152	0.0894	0.8442	0.7460	0.2582	0.1815	0.0551	0.9036	0.8482	0.2162
LIFT	**0.3361**	**0.0981**	**0.8496**	**0.7565**	**0.2724**	**0.1885**	**0.0582**	**0.9127**	**0.8612**	**0.2239**

Baselines. Retrieval baselines include YouTubeDNN (Covington et al., 2016), GRU4Rec (Hidasi et al., 2015), SASRec (Kang & McAuley, 2018), HSTU (Zhai et al., 2024), and UniPinRec (Li et al., 2026). Ranking baselines include DeepFM (Guo et al., 2017), DIN (Zhou et al., 2018), DCNv2 (Wang et al., 2021), RankMixer (Zhu et al., 2025), HSTU, and UniPinRec. Joint baselines include SharedBottom (Caruana, 1997), MMoE (Ma et al., 2018), PLE (Tang et al., 2020), HSTU, and UniPinRec. OneRec and OnePiece represent generative and latent-reasoning formulations, respectively, and are discussed in Related Work. Our direct comparison focuses on methods evaluated under the common catalog-based retrieval–ranking protocol.

Within each task, all methods use the same chronological splits, candidate construction, supervision, and evaluation protocol. Sequential models use the same maximum interaction history, with hidden dimension and depth aligned where applicable, while retaining their native input interfaces and prediction heads. Checkpoints are selected by validation NDCG@100 for retrieval, AUC for ranking, and Joint Score for joint training; the test set is used only for final evaluation. Unless otherwise specified, LIFT uses 100 historical interactions, three Transformer layers, hidden dimension 256, and eight attention heads. All reported quality metrics are averaged over random seeds 2026, 2027, and 2028.

Overall Results. Table 2 shows that LIFT's Joint Score reaches 0.2724 on ML-20M, improving over HSTU by 4.9%, and 0.2239 on Taobao, improving over UniPinRec by 3.6%. Dedicated retrieval and ranking results are reported in Appendix C.

4.2 Retrieval–Ranking Pareto Frontier

To examine retrieval–ranking trade-offs, for each joint model, we vary λ_{ret} over $\{0.0078125, 0.015625, 0.03125, 0.0625, 0.125, 0.25, 0.5, 1, 2\}$ while fixing $\lambda_{\text{rank}} = 1$ on ML-20M. Each point in Figure 3 reports the test performance of the checkpoint selected by validation Joint Score, and the connected non-dominated points form each method's empirical Pareto frontier.

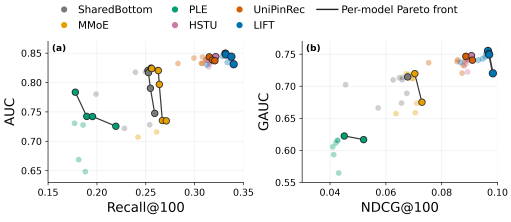


Figure 3: Retrieval–ranking trade-offs on ML-20M. (a) Recall@100 versus AUC; (b) NDCG@100 versus GAUC. Colors denote different methods, and black line segments connect each method's non-dominated configurations. Higher values are better on both axes.

Increasing λ_{ret} shifts LIFT toward retrieval-oriented operating points, while moderate weights preserve stronger ranking quality. Across the evaluated loss weights, LIFT configurations form the global empirical Pareto frontier in both Recall@100–AUC and NDCG@100–GAUC spaces, indicating stable retrieval–ranking trade-offs across different operating points.

4.3 Ablation Studies

Component Ablations. Table 3 uses a 2×2 design to isolate Role-Conditioned Attention and Pre-LN Bias while keeping the lifecycle sequence, task readouts, heads, and training protocol fixed.

Table 3: Component ablations on ML-20M. Higher values are better for all metrics.

Variant	R@100	N@100	AUC	GAUC	Joint Score
Basic Transformer	0.3255	0.0941	0.8464	0.7535	0.2663
+ Pre-LN Bias	0.3330	0.0963	0.8463	0.7541	0.2695
+ Role-Cond. Attn.	0.3292	0.0947	0.8480	0.7537	0.2672
LIFT	**0.3361**	**0.0981**	**0.8496**	**0.7565**	**0.2724**

Removing Pre-LN Bias reduces Joint Score from 0.2724 to 0.2672, removing Role-Conditioned Attention reduces it to 0.2695, and removing both yields 0.2663. The full model performs best on all reported metrics, supporting complementary contributions from the two components.

Lifecycle Sequence-Structure Ablation. Table 4 compares alternative lifecycle groupings and task readouts under the same encoder and training protocol. Multiple letters within one pair of brackets denote a merged token, while `[]` denotes an explicit query token. For `[RICA]`, retrieval reads the preceding interaction state, while ranking reads the current event state with the target Action masked.

Table 4: Lifecycle sequence-construction ablations on ML-20M. Higher values are better for all metrics.

Sequence	Ret. Readout	Rank. Readout	R@100	N@100	AUC	GAUC	Joint Score
`[RICA]`	$[\text{RICA}]_{t-1}$	$[\text{RICA}]_{t}$	0.3152	0.0894	0.8442	0.7460	0.2582
`[RIC][A]`	$[\text{A}]_{t-1}$	`[RIC]`	0.3195	0.0921	0.8457	0.7533	0.2634
$[]_I$`[RIC][A]`	$[]_I$	`[RIC]`	0.3226	0.0931	0.8464	0.7532	0.2647
`[R][IC][A]`	`[R]`	`[IC]`	0.3325	0.0952	0.8419	0.7390	0.2652
$[]_I$`[RIC]`$[]_A$`[A]`	$[]_I$	$[]_A$	0.3219	0.0922	0.8488	0.7553	0.2639
`[R][I][C][A]`	**`[R]`**	**`[C]`**	**0.3361**	**0.0981**	**0.8496**	**0.7565**	**0.2724**

The fully merged `[RICA]` representation obtains a Joint Score of 0.2582, whereas the fully factorized `[R][I][C][A]` sequence reaches 0.2724. The alternative four-token layout $[]_I$`[RIC]`$[]_A$`[A]` reaches 0.2639. The gap between the two four-token layouts suggests that aligning token boundaries with information-availability stages is important for constructing effective retrieval and ranking states.

4.4 Capacity and Sequence Scaling Analysis

Figure 4 compares LIFT and UniPinRec under depth, width, and sequence expansion.

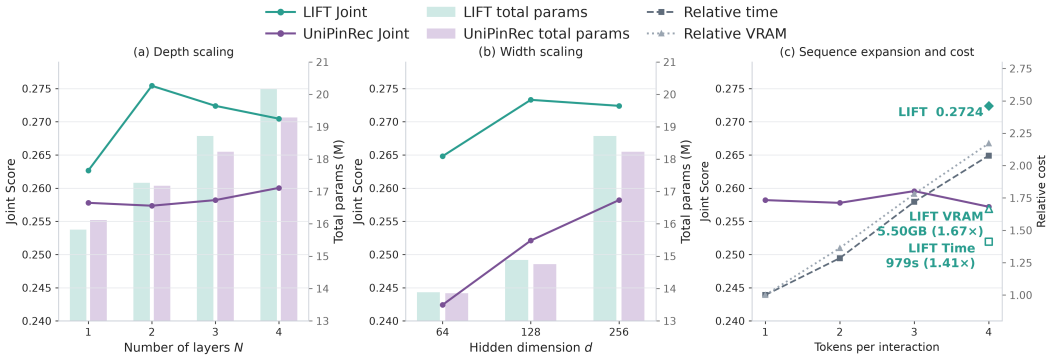


Figure 4: Capacity and sequence scaling on ML-20M. (a) Depth scaling. (b) Width scaling. (c) Sequence expansion and computational cost.

Capacity scaling. LIFT achieves a higher Joint Score than UniPinRec at every evaluated depth and hidden dimension. Notably, LIFT with $d = 64$ reaches 0.2648 using 13.88M parameters, exceeding UniPinRec with $d = 256$, which reaches 0.2582 with 18.23M parameters.

Sequence expansion. Adding blank tokens to UniPinRec leaves Joint Score nearly unchanged (0.2572–0.2596), whereas LIFT reaches 0.2724 with four semantic lifecycle tokens. At the matched four-token sequence length and the same encoder depth and hidden dimension, LIFT completes the 24-epoch training run in 979.0 seconds versus 1440.8 seconds for UniPinRec with three blank tokens, a 32.1% reduction, while peak memory decreases from 7.16 GB to 5.50 GB.

5 Conclusion

We presented LIFT, a unified retrieval–ranking model built around lifecycle-aware interaction factorization. LIFT decomposes each interaction into Request, Item, Context, and Action states according to information availability, allowing retrieval and ranking to share causal history modeling while reading representations aligned with their respective prediction stages. Role-conditioned attention and a lightweight Pre-LN Bias instantiate this factorization within a shared Transformer backbone. Experiments on ML-20M and Taobao show consistent improvements in joint retrieval–ranking quality, while loss-weight sweeps demonstrate favorable trade-offs between the two tasks. Lifecycle sequence ablations and scaling analyses further characterize the effects of sequence construction and model capacity.

AI USE STATEMENT

Generative AI tools were used during manuscript preparation for language polishing, consistency checking, code assistance, and feedback on experimental presentation. The authors independently verified all technical content, implementations, experimental results, analyses, and conclusions, and take full responsibility for the content of the paper.

REFERENCES

Alibaba Tianchi. User behavior data from taobao for recommendation. `https://tianchi.aliyun.com/dataset/649`, 2017. Accessed: 2026-09-26.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pp. 7–10, 2016.

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

Sunhao Dai, Jiakai Tang, Jiahua Wu, Kun Wang, Yuxuan Zhu, Bingjun Chen, Bangyang Hong, Yu Zhao, Cong Fu, Kangle Wu, et al. Onepiece: Bringing context engineering and reasoning to industrial cascade ranking system. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 7117–7128, 2026.

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

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Weihao Gao, Xiangjun Fan, Chong Wang, Jiankai Sun, Kai Jia, Wenzhi Xiao, Ruofan Ding, Xingyan Bin, Hui Yang, and Xiaobing Liu. Deep retrieval: learning a retrievable structure for large-scale recommendations. *arXiv preprint arXiv:2007.07203*, 2020.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

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

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, pp. 263–272. Ieee, 2008.

Xu Huang, Defu Lian, Jin Chen, Liu Zheng, Xing Xie, and Enhong Chen. Cooperative retriever and ranker in deep recommenders. In *Proceedings of the ACM Web Conference 2023*, pp. 1150–1161, 2023.

Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. In *IJCAI*, volume 19, pp. 2592–2599, 2019.

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

Hanyu Li, Yi-Ping Hsu, Aditya Mantha, Prabhat Agarwal, Laksh Bhasin, Jialu Wang, Hongtao Lin, Bella Huang, Yaxin Li, Xinyi Li, et al. Unipinrec: Unifying generative retrieval and ranking at pinterest scale. *arXiv preprint arXiv:2606.00422*, 2026.

Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*, pp. 322–330, 2020.

Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1754–1763, 2018.

Weiwen Liu, Yunjia Xi, Jiarui Qin, Fei Sun, Bo Chen, Weinan Zhang, Rui Zhang, and Ruiming Tang. Neural re-ranking in multi-stage recommender systems: A review. *arXiv preprint arXiv:2202.06602*, 2022.

Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1930–1939, 2018.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Li-Wei Pan, Wei-Ke Pan, Mei-Yan Wei, Hong-Zhi Yin, and Zhong Ming. A survey on sequential recommendation. *Frontiers of Computer Science*, 20(3):2003606, 2026.

Jiarui Qin, Jiachen Zhu, Bo Chen, Zhirong Liu, Weiwen Liu, Ruiming Tang, Rui Zhang, Yong Yu, and Weinan Zhang. Rankflow: Joint optimization of multi-stage cascade ranking systems as flows. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 814–824, 2022.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM computing surveys (CSUR)*, 51(4):1–36, 2018.

Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pp. 995–1000. IEEE, 2010.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian Tang. Autoint: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1161–1170, 2019.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 1441–1450, 2019.

Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM conference on recommender systems*, pp. 269–278, 2020.

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

Jizhe Wang, Pipei Huang, Huan Zhao, Zhibo Zhang, Binqiang Zhao, and Dik Lun Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 839–848, 2018.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the web conference 2021*, pp. 1785–1797, 2021.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pp. 3597–3606, 2020.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International conference on machine learning*, pp. 10524–10533. PMLR, 2020.

Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM conference on recommender systems*, pp. 269–277, 2019.

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

Luankang Zhang, Kenan Song, Yi Quan Lee, Wei Guo, Hao Wang, Yawen Li, Huifeng Guo, Yong Liu, Defu Lian, and Enhong Chen. Killing two birds with one stone: Unifying retrieval and ranking with a single generative recommendation model. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2224–2234, 2025.

Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1059–1068, 2018.

Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 5941–5948, 2019.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pp. 1893–1902, 2020.

Han Zhu, Xiang Li, Pengye Zhang, Guozheng Li, Jie He, Han Li, and Kun Gai. Learning tree-based deep model for recommender systems. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1079–1088, 2018.

Jie Zhu, Zhifang Fan, Xiaoxie Zhu, Yuchen Jiang, Hangyu Wang, Xintian Han, Haoran Ding, Xinmin Wang, Wenlin Zhao, Zhen Gong, et al. Rankmixer: Scaling up ranking models in industrial recommenders. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 6309–6316, 2025.

A Dataset Feature Assignment

Table 5 summarizes the lifecycle-role assignment used in our experiments. Derived historical statistics are computed causally using only information available before the current interaction. The target Action is excluded from the input at the prediction position and is used only to construct supervision.

Table 5: Lifecycle feature assignment used in ML-20M and Taobao.

Dataset	Request	Item	Context	Action
ML-20M	weekend user activity time since last	item ID release year genres	item count (log) average rating	rating positive label
Taobao	user activity (log2) time gap	item ID category	item count (log) user–item count	behavior type purchase label

For ML-20M, the positive ranking label is defined as a rating greater than 4. For Taobao, purchase is treated as positive feedback and the other behavior types as negative feedback. Item-count, user-activity, user–item-count, average-rating, and time-gap features are constructed from historical observations only. Sequence-based models do not expose the current target Action to the encoder.

B Hyperparameter Search Space

Table 6 summarizes the hyperparameter values considered during model selection. We use coordinate search and select configurations according to validation performance.

Table 6: Hyperparameter search spaces used in the experiments.

Task	Hyperparameter	Search space
Retrieval	Learning rate	$\{1e{-}4,\ 3e{-}4,\ 1e{-}3,\ 2e{-}3,\ 3e{-}3\}$
	Dropout	$\{0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5\}$
	Weight decay	$\{0.0,\ 0.001,\ 0.01\}$
	Temperature	$\{0.05,\ 0.07,\ 0.10\}$
	LogQ α	$\{0.50,\ 0.75,\ 1.00\}$
Ranking	Learning rate	$\{1e{-}4,\ 3e{-}4,\ 1e{-}3,\ 2e{-}3,\ 3e{-}3\}$
	Dropout	$\{0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5\}$
	Weight decay	$\{0.0,\ 0.001,\ 0.01\}$
Joint	Learning rate	$\{1e{-}4,\ 3e{-}4,\ 1e{-}3,\ 2e{-}3,\ 3e{-}3\}$
	Dropout	$\{0.0,\ 0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5\}$
	Weight decay	$\{0.0,\ 0.001,\ 0.01\}$
	Temperature	$\{0.05,\ 0.07,\ 0.10\}$
	LogQ α	$\{0.50,\ 0.75,\ 1.00\}$

C Dedicated Single-Task Results

We additionally evaluate all methods under dedicated retrieval and ranking training. Table 7 reports the results on ML-20M and Taobao.

Table 7: Dedicated retrieval and ranking results on ML-20M and Taobao.

(a) Retrieval

	ML-20M		Taobao	
Method	R@100	N@100	R@100	N@100
YouTubeDNN	0.2215	0.0554	0.0685	0.0191
GRU4Rec	0.3142	0.0909	0.1570	0.0466
SASRec	0.3110	0.0902	0.1609	0.0479
HSTU	0.3142	0.0889	0.1684	0.0511
UniPinRec	0.3186	0.0916	0.1694	0.0510
LIFT	**0.3352**	**0.0961**	**0.1790**	**0.0544**

(b) Ranking

	ML-20M		Taobao	
Method	AUC	GAUC	AUC	GAUC
DeepFM	0.7116	0.6851	0.8907	**0.8692**
DIN	0.7200	0.7061	0.8894	0.8664
DCNv2	0.7087	0.6937	0.8838	0.8645
RankMixer	0.8170	0.7088	0.9006	0.8361
HSTU	0.8480	0.7530	0.9070	0.8588
UniPinRec	0.8493	0.7545	0.9086	0.8493
LIFT	**0.8509**	**0.7569**	**0.9129**	0.8587

LIFT achieves the strongest retrieval results on both datasets and the highest ranking AUC on both. It also obtains the highest GAUC on ML-20M; on Taobao, DeepFM achieves the highest GAUC, while

LIFT retains the highest AUC. These results show that lifecycle factorization remains effective when retrieval and ranking are trained independently.

D CAUSAL MASK WITH MULTIPLE EXPOSURES

In practical recommendation logs, a single request event may contain multiple exposure samples corresponding to different candidate items (Wu et al., 2020; Ie et al., 2019). These candidates are generated under the same request and have no temporal precedence relation with each other. Directly applying a standard lower-triangular mask would therefore introduce an artificial ordering between co-occurring candidates. We extend the causal mask to preserve temporal causality across events while isolating different exposures within the same event. The formulation in the main text corresponds to the single-exposure case.

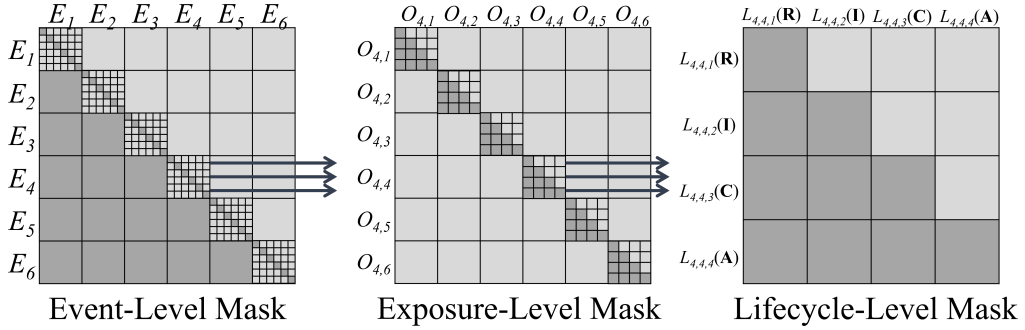


Figure 5: Causal mask construction for events containing multiple exposure samples. Visibility is determined by event-level temporal causality, exposure-level isolation, and lifecycle-level ordering.

Let q and k denote the query and key token positions, respectively. Each token position is associated with three indices:

$$(e_q, o_q, c_q), \qquad (e_k, o_k, c_k), \tag{16}$$

where e denotes the event index, o denotes the exposure index within an event, and c denotes the lifecycle-stage index. We assign

$$c(R) = 1, \qquad c(I) = 2, \qquad c(C) = 3, \qquad c(A) = 4, \tag{17}$$

following the lifecycle order

$$\text{Request} \rightarrow \text{Item} \rightarrow \text{Context} \rightarrow \text{Action}.$$

For clarity, the masks in this appendix are binary visibility masks. For any mask $M^*_{q,k} \in \{0, 1\}$, $M^*_{q,k} = 1$ indicates that query position q is allowed to attend to key position k. We use $\mathbb{I}[\cdot]$ to denote the indicator function.

Event-level mask. Cross-event attention follows temporal causality. A token may attend to all tokens belonging to earlier events, but not to future events:

$$M^E_{q,k} = \mathbb{I}[e_k < e_q]. \tag{18}$$

Exposure-level mask. Different exposures within the same event correspond to co-occurring candidate items. Since these candidates do not have a temporal ordering, tokens from one exposure cannot attend to tokens from another exposure within the same event. Visibility within the current event is therefore restricted to the same exposure:

$$M^O_{q,k} = \mathbb{I}[e_k = e_q] \wedge \mathbb{I}[o_k = o_q]. \tag{19}$$

Lifecycle-level mask. Within the same exposure, attention follows the lifecycle order. A token may attend only to lifecycle stages that are available no later than its current stage:

$$M^L_{q,k} = \mathbb{I}[c_k \leq c_q]. \tag{20}$$

Final attention mask. The final visibility relation combines the three constraints:

$$M_{q,k}^{\mathrm{Attn}} = M_{q,k}^{E} \vee \left(M_{q,k}^{O} \wedge M_{q,k}^{L}\right). \tag{21}$$

Thus, a token can attend to all tokens from earlier events. Within the current event, it can attend only to tokens from the same exposure whose lifecycle stages do not succeed its current stage. Tokens belonging to different exposures of the same event remain mutually invisible.

To match the additive attention mask used in the main model, the binary visibility mask is converted as

$$\widetilde{M}_{q,k} = \begin{cases} 0, & M_{q,k}^{\mathrm{Attn}} = 1, \\ -\infty, & M_{q,k}^{\mathrm{Attn}} = 0. \end{cases} \tag{22}$$

When every event contains exactly one exposure, i.e., $o_q = o_k$ for tokens within the same event, the exposure-level constraint becomes trivial. Under the event-major lifecycle serialization used by LIFT, Eq. 21 then reduces to the standard lower-triangular causal mask:

$$M_{q,k}^{\mathrm{Attn}} = \mathbb{I}[k \le q]. \tag{23}$$