



arXiv is now an independent nonprofit! [Learn more](#) ×



Why
HTML?

 Report
Issue

 Back to
Abstract

 PDF
Download
PDF



SlimPer: Make Personalization Model Slim and Smart

Siqi Wang*, Xianjie Chen*[†], Shaofeng Deng, Albert Chen, Romil Shah, Jiawei Huang, Zhaoqin Wang, Zhang Zhang, Yiqun Liu, Meilei Jiang, Anish Dubey, Moyan Mei, Tongxin Wang, Nathan Berrebbi, Misael Manjarres, Armand Sauzay, Shardul Kothapalli, Aryaman Vinchhi, Kevin Johnstone, Juheon Lee, Gufan Yin, Ziheng Huang, Justin Lin, Mert Terzihan, Yilin Qi, Cynthia Yang, Colin Pepler, Qi Ding, Ruohan Sun, Ge Song, Litao Deng, Parichay Kapoor, Matt Ma, Huihui Cheng, Jiyuan Zhang, Yanli Zhao, Yiping Han, Fangqiu Han, Ning Yao, Arun Singh, Jordan Edwards, Zhengyu Su, Abhishek Kumar, Guangdeng Liao, Ankit Asthana

Meta Platforms, Inc. USA

Abstract.

Transformer-style architectures are increasingly adopted for industrial recommendation systems, yet they inherit a design premise misaligned with the task: generative models rely on per-token autoregressive prediction, which justifies maintaining large intermediate tensors that scale with sequence length. In contrast, recommendation systems produce a single set of relevance scores for each $\langle \text{user}, \text{item} \rangle$ pair without token-level supervision. Leveraging this observation, we propose **SlimPer**, which reformulates personalized ranking as *iterative refinement of a compact, unified $\langle \text{user}, \text{item} \rangle$ knowledge base*. At each layer, the model selectively queries raw multi-modal user-side tokens, computes explicit relevance matching scores, and refines the knowledge base, all in $\mathcal{O}(N)$ per-layer cost with a fixed-size intermediate representation. As a result, model depth is decoupled from user history length, enabling deeper relevance understanding without proportional growth in compute or memory; request-only optimization further trims memory by sharing a single copy of user-side tokens across all candidate items. SlimPer unifies sparse, dense, and sequence features within a single backbone and provides inherent interpretability through its attention mechanism. Deployed on Instagram Reels and Feed, SlimPer yields measurable improvements in user engagement while streamlining the overall system and enabling effective modeling of 10k+ fine-grained user history events.

†

†conference: Preprint; ;

†

†booktitle: Preprint

†

†ccs: Information systems Recommender systems

†

†*Equal contribution. † Project lead and corresponding author (cxj@meta.com).

1. Introduction

1.1. The Transformer Mismatch in Recommendation

Discriminative personalization models, commonly known as ranking models, shape the content feeds, video recommendations, and product suggestions that users interact with every day. These models need to rank candidate items according to user preferences in real time while maintaining accurate predictions under production cost constraints.

Industrial recommendation models typically consume multi-modal input features, where each modality offers different trade-offs between computational cost and the ability to capture fine-grained information (see Table 1 for a detailed comparison). In common practice, different modalities are processed by separate specialized components and then shallowly stitched together for final predictions. Recently, motivated by the success of Transformers in large language models (LLMs), adapting transformer-like architectures to recommendation systems has become a popular direction, either for modeling variable-length sequence features (e.g., HSTU (Zhai et al., 2024), Interformer (Zeng et al., 2025)) or for unifying the modeling of different feature modalities (e.g., OneTrans (Zhang et al., 2025), HHFT (Yu et al., 2025), RankMixer (Zhu et al., 2025)).

However, this trend overlooks a structural difference between recommendation systems and LLMs. In LLMs, per-position autoregressive supervision requires maintaining full token-level representations throughout the network, and the computational cost is amortized across predictions at every token position. *Current recommendation systems have no such requirement:*

they are trained for discriminative prediction and produce a single set of relevance scores for each $\langle user, item \rangle$ pair without token-level autoregressive losses. As a result, Transformer-like designs face an unnecessary dilemma: either intermediate representations are heavily compressed across layers, leading to information loss, or large intermediate tensors are preserved, incurring considerable memory and computational overhead as the number of input tokens scales. This dilemma is not inherent to the recommendation itself; it is an artifact of borrowing an architectural premise that does not apply.

1.2. A New Paradigm: Iterative Refinement of a Knowledge Base

We argue that the appropriate abstraction for discriminative recommendation is not sequence modeling, but *iterative refinement* over a compact user-item knowledge base. Under this formulation, the model maintains a fixed-size representation of $\langle user, item \rangle$ relevance and progressively refines it by selectively querying raw multi-modal evidence, without materializing large intermediate tensors whose size grows with input sequence length.

Based on this principle, we propose **SlimPer**, a recommendation-centric architecture for personalized ranking. As shown in Fig. 1, SlimPer formulates personalized ranking as an iterative refinement process over a unified fixed-size $\langle user, item \rangle$ knowledge base. Each layer performs three steps: (1) *Selection*: all-modality-aware QKV attention queries the complete set of raw user-side tokens conditioned on the current knowledge base; (2) *Matching*: explicit dot-product relevance scores are computed between the retrieved evidence and multifaceted templates; (3) *Refinement*: the knowledge base is updated using the matched evidence.

This design resolves the transformer dilemma at its root: increasing model depth to improve understanding of the $\langle user, item \rangle$ pair no longer requires maintaining large intermediate tensors whose size scales with the number of user-side input tokens. At the same time, model quality is preserved because every layer retains direct access to the complete set of raw user-side information when refining the knowledge base. The resulting smaller intermediate representations translate directly into lower computational and memory cost (i.e., *Slim*). Moreover, the knowledge base focuses its entire learning capacity on $\langle user, item \rangle$ relevance, allocating resources more deliberately toward relevance matching compared to architectures that must maintain full token-level representations (i.e., *Smart*).

In addition, inspired by the success of request-only optimization (ROO) in HSTU (Zhai et al., 2024) for processing user-side sequence features, we explicitly separate user-side features from non-user-side features and extend ROO to all user-side feature modalities. This is an

other recommendation-centric design choice, motivated by the fact that recommendation models typically rank multiple candidate items for the same user during both training and inference.

We conduct experiments on Instagram’s two major ranking surfaces, Reels and Feed, using final-stage ranking models. SlimPer broadly outperforms prior models while achieving lower computational and memory cost with a more streamlined model architecture.

1.3. Contributions

In summary, our main contributions are as follows:

- (1) **Iterative refinement as a new paradigm for personalized ranking:** We formulate discriminative recommendation as an iterative refinement process over a compact $\langle user, item \rangle$ knowledge base, departing from the dominant transformer-mimicking paradigm. At each refinement step, the model directly queries the full set of raw multi-modal tokens, thereby decoupling model depth from input sequence length. This design achieves $\mathcal{O}(N)$ per-layer complexity with a fixed-size intermediate representation, effectively resolving the efficiency-quality trade-off faced by transformer-like architectures when scaling to long user histories.
- (2) **Unified multi-modal architecture with recommendation-centric efficiency:** SlimPer naturally provides a unified backbone for processing sparse, dense, and sequence features using only standard neural network building blocks including MLPs, QKV attention, linear projections, and RMSNorm (Zhang and Sennrich, [2019](#)). Furthermore, inspired by the effectiveness of request-only optimization (ROO) in HSTU for user history modeling, SlimPer extends ROO awareness to all user-side feature modalities: user-side tokens are computed once per request batch and shared across all candidate items within the same request, reducing redundant computation and memory overhead. Together, these design choices improve both ranking quality and system efficiency compared with transformer-mimicking baselines.
- (3) **Intrinsic interpretability for real-world recommender systems:** The iterative refinement framework also provides intrinsic interpretability. At each layer, the QKV attention mechanism directly associates recommendation outcomes with specific user interaction history events, making model behavior easier to interpret and debug in real-world recommendation scenarios.

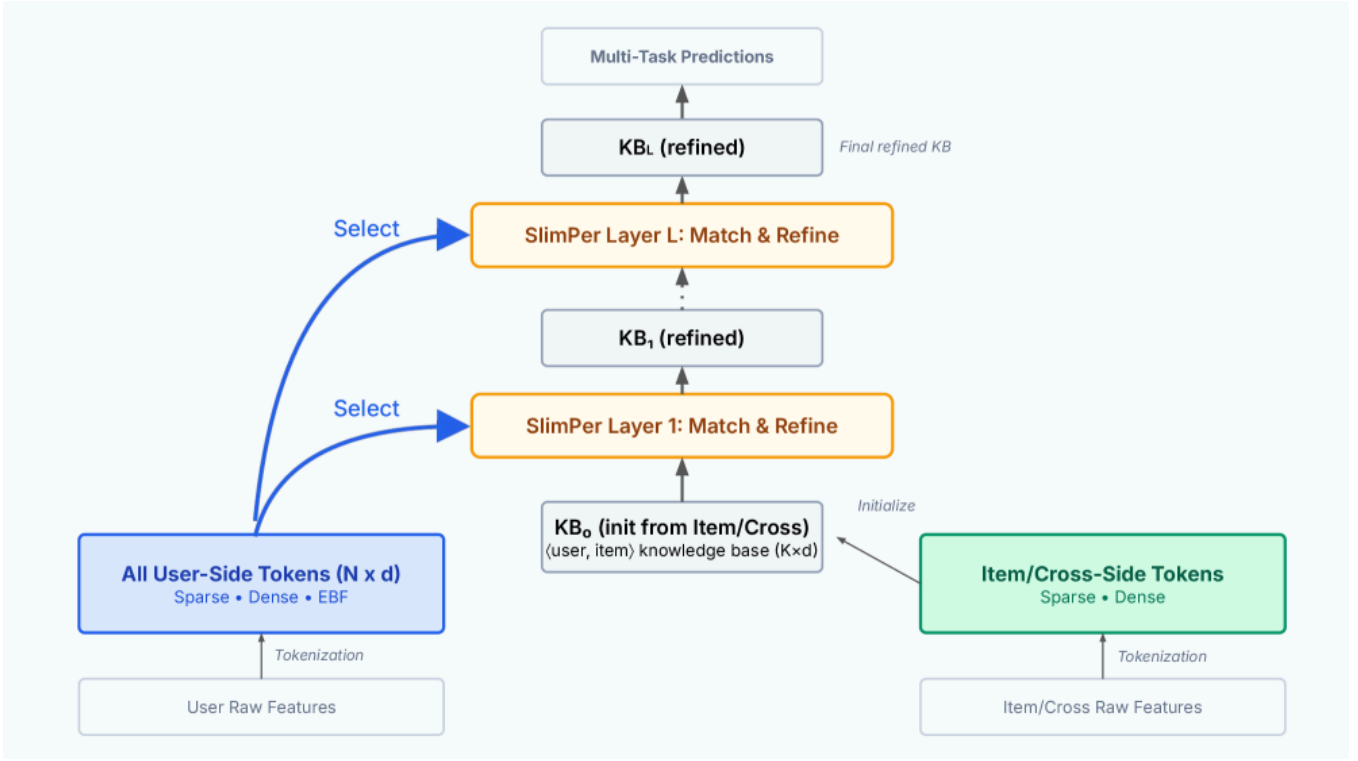


Figure 1. **High-level overview of SlimPer.** Three design principles: **(1) Slim** — fixed-size knowledge base ($K \times d$), cost decoupled from input length N ; **(2) Complete Access** — user-side tokens queried at every layer; **(3) ROO-Aware** — user-side tokens computed once and shared across all candidates in a request.

2. Methodology

2.1. Design Philosophy

The design principles motivating SlimPer arise from the observation that recommendation requires only discriminative prediction, not per-token autoregressive output. The idea of iterative refinement over a compact knowledge base, and the resulting architectural advantages (i.e., decoupled depth and length, complete information access, and more focused capacity allocation), were introduced in Section 1. We now formalize these principles into a concrete architecture.

2.2. Problem Formulation

Consider a recommendation request consisting of a user u and a set of candidate items $\mathcal{I} = \{i_1, i_2, \dots, i_C\}$ to be ranked. The input consists of diverse sets of feature modalities: a set

of Sparse features (**S**), Dense features (**D**), and a sequence of user interaction history events (**E**). The goal is to learn a scoring function $f(u, i; \mathbf{S}, \mathbf{D}, \mathbf{E}) \rightarrow \mathbb{R}^\tau$ that estimates the relevance of i by matching it against the user’s latent interests representation. In multi-task settings, this function simultaneously predicts τ objectives representing different engagement signals.

Table 1. Different information modalities

Modality	Input Type	Representative Signals
Sparse	High-cardinality categorical IDs	Summarized user interests (e.g., top engaged creators or categories), providing salient but coarse-grained preference signals.
Sequence	Structured user engagement records	Fine-grained behavioral signals capturing individual user actions (e.g., content consumed, dwell time, engagement type, and timestamp), providing high-resolution temporal information.
Dense	Continuous numerical values	Contextual and real-time signals, such as user-level statistics (e.g., CTR), item quality metrics (e.g., reshare rate), and contextual attributes (e.g., time or device).

2.3. Notation and Preliminaries

We restrict the model to a minimal set of building blocks: **MLP** (multilayer perceptron with SiLU activations), $\mathcal{L}$ (segment-wise linear projection: $\beta = \mathcal{L}(\alpha) = \rho\alpha$, $\rho \in \mathbb{R}^{L_{\text{out}} \times L_{\text{in}}}$), and the all-modality $\langle user, item \rangle$ knowledge base $\mathcal{X}$, initialized as $\mathcal{X}^0 = \mathcal{L}(\mathbf{S}_{\text{in}})$ from non-user-side input tokens. The knowledge base has shape $\mathcal{X}^k \in \mathbb{R}^{K \times d}$, where K is the number of knowledge base slots and d is the embedding dimension. The size may in general differ across layers; in our experiments we keep it consistent with $K = 64$ across all layers. Full notation details are in Appendix B.

2.4. Overall Architecture

A SlimPer model consists of stacked SlimPer layers preceded by multi-modality tokenization. The overall architecture is shown in Fig. 2. The model takes raw sparse and dense features, along with a sequence of user history events, as inputs. All features are first tokenized using their corresponding tokenizers (see Table 5), producing a set of multi-head d -dimensional embeddings. For simplicity, we describe our algorithms in single-head form without loss of

generality. Each layer performs the Select–Match–Refine cycle (Section 2.6), leveraging standard Query–Key–Value (QKV) attention for all–modality–aware token selection and explicit relevance matching.

The model inherently incorporates request–only optimization (ROO) awareness by explicitly decoupling user–side and item–side features. Historically, ROO has been associated with quality–efficiency trade–offs (e.g., two–tower architectures in early–stage ranking) or narrowly applied to user history modeling. Our design extends ROO awareness across the entire model, encompassing all user–side input features beyond just user history. Tokenization for all user–side features is performed once per request batch, and a single copy of these user–side tokens is shared across all items within the same request, minimizing memory and compute cost.

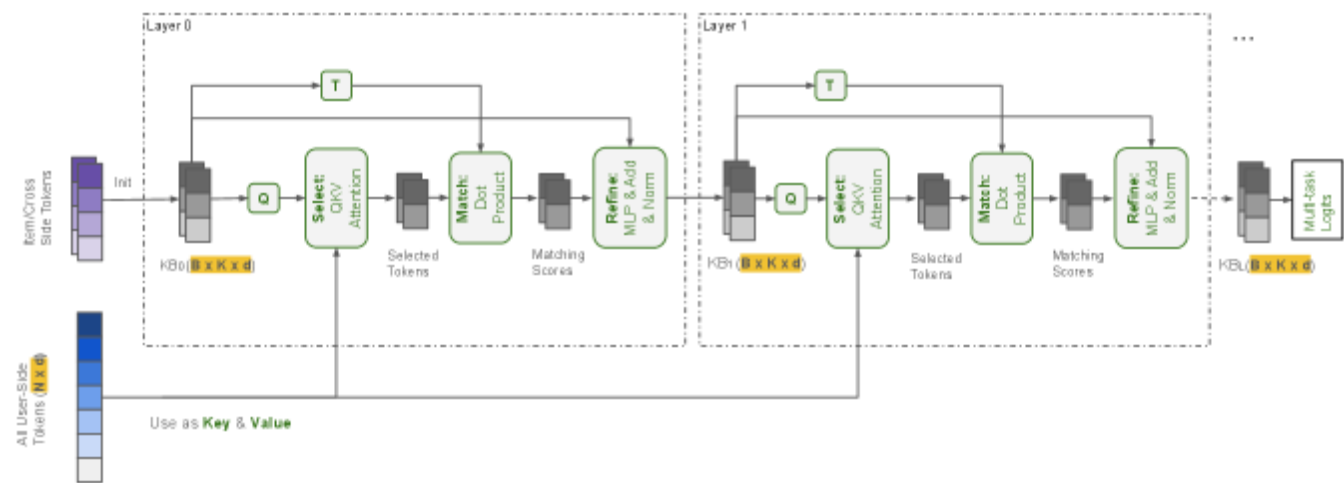


Figure 2. Architecture of the multi-layer SlimPer model. Each layer performs iterative refinement of a compact knowledge base ($K=16\text{--}64$ slots) by querying the full set of user-side tokens ($N=\mathcal{O}(10^3)\text{--}\mathcal{O}(10^4)$ in production), decoupling the memory and compute cost of stacking layers from input sequence length.

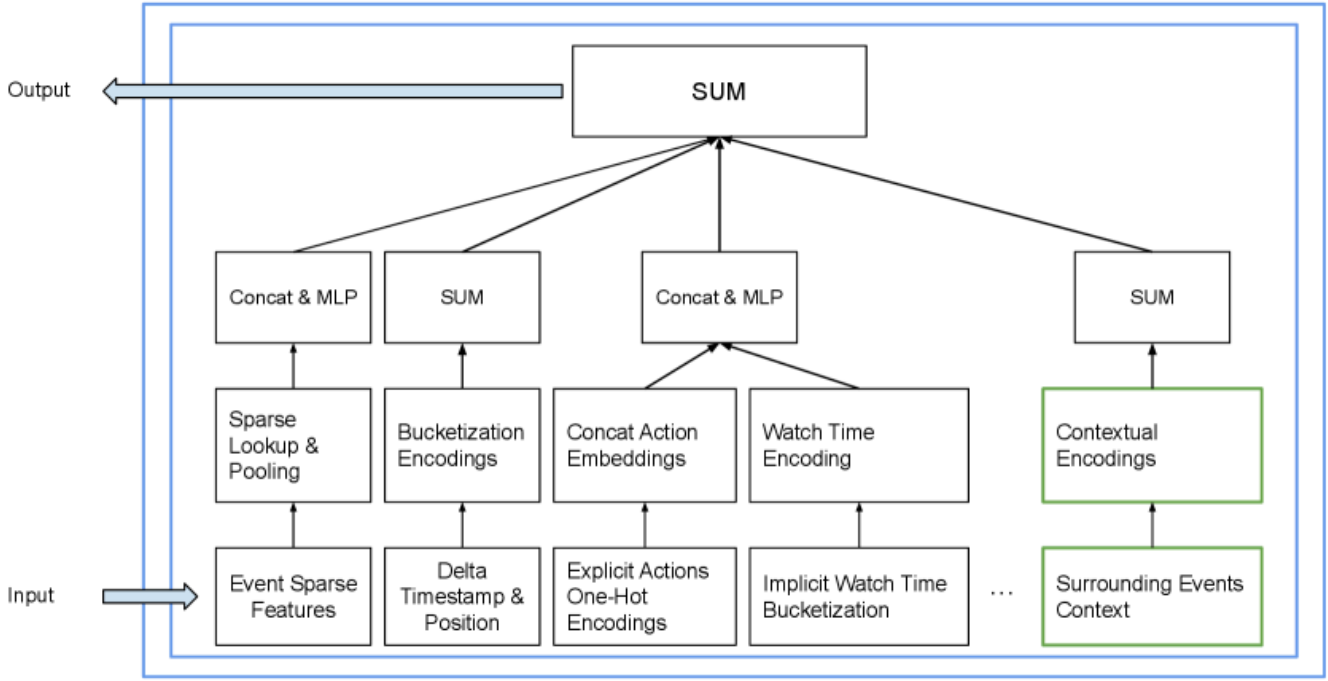


Figure 3. Event modeling module that encodes each event using content, engagement type, temporal and contextual information.

2.5. Tokenization Module

All raw input features are converted into modality-specific tokens within the unified backbone. Sparse features are tokenized via embedding pooling into fixed-size vectors. Each user history event is encoded into a $1 \times d$ token by fusing content, engagement type, temporal, and contextual information. Dense features are concatenated and projected through an MLP. The resulting multi-modal token set serves as input to the SlimPer layers. Full tokenization details for standard components (embedding pooling and dense processing) are provided in Appendix C.

Contextual Encoding for Event Tokens.

Intuitively, user behavior often exhibits local continuity, where an event is most informative when interpreted together with its nearby events. To capture these local contextual relationships, we introduce an additional contextual encoding $\mathbf{e}_{\text{context}}^{(t)}$ that explicitly summarizes neighborhood information around each event token (Fig. 3, green box):

$$(1) \quad \mathbf{e}_{\text{context}}^{(t)} = \Psi_{\text{context}}(x_{t-w+1}, \dots, x_t)$$

where Ψ_{context} represents the encoding function, w is the window size, and $\mathbf{x}_t$ denotes the information collected from the user history event at position t . We explored convolutional neural networks (LeCun et al., 1998), sliding-window attention (Beltagy et al., 2020), and combinations of both as Ψ_{context} ; all variants achieved similar performance. We also experimented with using different information at each event position for contextual encoding; see the ablation study in Section 4.5.1 for details.

2.6. SlimPer Layer: Select, Match, and Refine

As illustrated in Fig. 2, each SlimPer layer follows a three-step iterative refinement procedure. Tokens from all modalities are jointly processed in Steps 1 and 2, where QKV attention is used to retrieve relevant user-side tokens and compute explicit relevance matching scores. Step 3 then refines the knowledge base with the matched evidence. This design re-allocates compute and memory away from pairwise user-side token-to-token interactions and toward learning discriminative user-item interactions. Table 2 compares the compute and memory allocation of SlimPer with representative transformer-like architectures.

Table 2. Per-item Big-O complexity across architectures. The rightmost column is per-layer user-item interaction capacity, where SlimPer’s $K\times$ advantage is analyzed in Section 2.7. Typical production values: $N=\mathcal{O}(10^3)\text{--}\mathcal{O}(10^4)$, $K=64$, $q=16$, $B\approx 5$ (training) / $\mathcal{O}(10^2)$ (inference).

Model	Tokenization (per item)	Cost Stacked Layers item)	Cost (per User-Item Interaction Capacity
Conventional DLRM Model • <i>not ROO-aware; no sequence feature support</i>	Memory: $\mathcal{O}(N)$ Compute: $\mathcal{O}(N)$	Memory: $\mathcal{O}(L \cdot 64)$ Compute: $\mathcal{O}(L \cdot N \cdot 16)$	$\mathcal{O}(L \cdot N \cdot 64)$
ROO-aware Transformer-like Model (e.g., HSTU) • <i>ROO-aware for user history modeling</i>	Memory: $\mathcal{O}(N / B)$ Compute: $\mathcal{O}(N / B)$	Memory: $\mathcal{O}(L \cdot N / B)$ Compute: $\mathcal{O}(L \cdot (N+1)^2 / B)$	$\mathcal{O}(L \cdot N)$
Non-ROO-aware Transformer-like Model (e.g., OneTrans) • <i>not ROO-aware</i>	Memory: $\mathcal{O}(N)$ Compute: $\mathcal{O}(N)$	Memory: $\mathcal{O}(L \cdot N)$ Compute: $\mathcal{O}(L \cdot N^2)$	$\mathcal{O}(L \cdot N)$
Sub-quadratic Transformer-like Model (e.g., linear or sliding-window attention) • <i>linearized attention</i>	Memory: $\mathcal{O}(N)$ Compute: $\mathcal{O}(N)$	Memory: $\mathcal{O}(L \cdot N)$ Compute: $\mathcal{O}(L \cdot N)$	$\mathcal{O}(L \cdot N)$

SlimPer

- *ROO-aware for all feature modalities*

Memory: $\mathcal{O}(N / B)$

Compute:

 $\mathcal{O}(N / B)$ Memory: $\mathcal{O}(L \cdot K)$ Compute: $\mathcal{O}(L \cdot N \cdot q)$ $\mathcal{O}(L \cdot N \cdot K)$ **2.6.1. Step 1: Token Selection via All-Modality-Aware Attention**

We leverage information from all input feature modalities during attention-based token selection. Let $\mathcal{X}^k$ denote the unified knowledge base at iteration k , representing the model's current understanding of the user-item pair. Query vectors $\mathbf{Q}$ are derived from $\mathcal{X}^k$ through segment-wise linear projection:

$$(2) \quad \mathbf{Q} = \mathcal{L}(\mathcal{X}^k) \in \mathbb{R}^{q \times d}.$$

Intuitively, $\mathbf{Q}$ captures the information that the model seeks to retrieve from user-side tokens, conditioned on the current user-item context encoded in $\mathcal{X}^k$. All user-side input tokens serve as keys $\mathbf{K}$ and values $\mathbf{V}$ (with optional linear projection). We organize them into two key-value sets: fixed-size sparse features and variable-length sequence features. QKV attention is applied to both sets to compute attention weights, which are used to aggregate the corresponding values to produce attention output $\mathbf{R}$:

$$(3) \quad \mathbf{R} = \Phi(\mathbf{Q}, \mathbf{K}) \mathbf{V}$$

Here the attention kernel Φ is modality-dependent. For fixed-size sparse tokens, we employ a lightweight parametric kernel implemented as an **MLP** since the token count is constant. For variable-length user history sequence features, we adopt a non-parametric standard scaled dot-product kernel (Vaswani et al., 2017). Dense features, which typically have relatively small dimensionality (e.g., ~ 1000 floats), are directly propagated and fused at each layer without passing through an attention kernel:

$$(4) \quad \Phi(\mathbf{Q}, \mathbf{K}) = \begin{cases} \text{MLP}(\text{Concat}(\mathcal{L}(\mathbf{Q}), \mathcal{L}(\mathbf{K}))), & \text{sparse features} \\ \text{Softmax}(\gamma \mathbf{Q} \mathbf{K}^T), & \text{sequence features} \end{cases}$$

where γ denotes the softmax temperature. Applying this formulation to the two user-side token streams yields two attention-based representations:

$$(5) \quad \mathbf{R}_s = \Phi_s(\mathbf{Q}, \mathbf{S}) \mathbf{S}, \quad \mathbf{R}_s \in \mathbb{R}^{q \times d},$$

$$(6) \quad \mathbf{R}_e = \Phi_e(\mathbf{Q}, \mathbf{E}) \mathbf{E}, \quad \mathbf{R}_e \in \mathbb{R}^{q \times d}.$$

The attention outputs $\mathbf{R}_s$ and $\mathbf{R}_e$ summarize the evidence selected from both token streams. In our implementation, the raw sparse feature tokens $\mathbf{S}$ and sequence feature tokens $\mathbf{E}$ are directly used as keys and values in the QKV attention. We also explored applying additional linear projections to derive keys and values, but observed no performance improvement; therefore, these projections are omitted for computational efficiency.

This design yields an all-modality-aware attention mechanism. Since the query representation $\mathbf{Q}$ is derived from the unified knowledge base $\mathcal{X}^k$, it incorporates information from all input modalities. Consequently, sparse feature information can influence token selection over sequence features, and vice versa.

2.6.2. Step 2: Explicit Multifaceted Dot-Product Relevance Matching

To perform relevance matching, we derive a set of multifaceted templates $\mathbf{T} \in \mathbb{R}^{t \times d}$ from the unified $\langle user, item \rangle$ knowledge base using a linear transformation:

$$(7) \quad \mathbf{T} = \mathcal{L}(\mathcal{X}^k), \quad \mathbf{T} \in \mathbb{R}^{t \times d}.$$

The relevance matching scores are computed via explicit dot products between the templates and each stream's attention output:

$$(8) \quad \lambda_s = \text{DotProduct}(\mathbf{R}_s, \mathbf{T}), \quad \lambda_s \in \mathbb{R}^{q \times t},$$

$$(9) \quad \lambda_e = \text{DotProduct}(\mathbf{R}_e, \mathbf{T}), \quad \lambda_e \in \mathbb{R}^{q \times t}.$$

2.6.3. Step 3: Refine the Knowledge Base

The relevance scores are then normalized and transformed via MLP_μ to refine the $\langle user, item \rangle$ knowledge base $\mathcal{X}^k$ and produce the next layer's representation $\mathcal{X}^{k+1}$. The dense tokens $\mathbf{D}$ are concatenated to incorporate dense feature information when updating the knowledge base:

$$(10) \quad \mathcal{X}^{k+1} = \mathcal{X}^k + \text{MLP}_\mu \left(\text{Concat} \left(\text{RMSNorm}(\lambda_s), \text{RMSNorm}(\lambda_e), \mathcal{L}(\mathcal{X}^k), \mathbf{D} \right) \right)$$

At each layer, we optionally output a task embedding for each prediction task, representing the model’s current understanding of that task. The task embeddings across layers can be summed before applying a final linear projection to produce task logits:

$$(11) \quad \mathbf{P}_p^k = \text{MLP}_p(\mathcal{L}(\mathcal{X}^k))$$

where $p \in [1, \tau]$ indexes the prediction tasks.

2.7. Iterative Personalization via Stacking SlimPer Layers

SlimPer is inherently stackable: multiple SlimPer layers are composed sequentially, each performing one iteration of the Select—Match—Refine cycle (Section 2.6) over the shared knowledge base $\mathcal{X}$. In our production setting, we typically stack 5 to 10 layers.

The intuition is that **deeper layers benefit from the evidence accumulated by earlier layers**, enabling progressively more informed personalization. For example, earlier layers may focus on collecting broad evidence related to the target item, such as whether the user historically engaged with dog videos. This accumulated evidence then conditions subsequent layers, which can focus on more nuanced signals, e.g., whether recent behaviors (e.g., frequent skipping of dog videos) indicate that the user’s interest has shifted. In this way, later layers can determine whether earlier preferences should be reinforced or discounted.

This iterative refinement capability is enabled by two architectural properties. First, each layer directly attends to the *complete set of raw input tokens* across all modalities, preserving fine-grained signals without incurring the memory overhead of maintaining large intermediate tensors. Unlike transformer-like architectures, which progressively compress information into intermediate hidden states, SlimPer performs lossless cross-modality and cross-side (user $\leftrightarrow$ item) fusion at every depth. Second, the knowledge base maintains a fixed size throughout the refinement process. As a result, increasing depth does not require intermediate memory to scale with input length, making it practical to stack many refinement layers even for users with long interaction histories.

2.7.1. Information–Theoretic Justification

The fixed–size knowledge base that enables these efficiency gains may appear to sacrifice information, but analysis reveals the opposite: SlimPer’s bottleneck *redirects* capacity toward task–relevant signals rather than discarding them.

The bottleneck is capacity–abundant. The recommendation task demands only τ scalar predictions per *user, item* pair, so the mutual information between target $\mathbf{Y}$ and input $\mathbf{X}$ is bounded by $H(\mathbf{Y}) \leq \tau$ bits—far below the representational capacity of a $\mathbf{K} \times \mathbf{d}$ knowledge base. Moreover, as shown in the interaction capacity column of Table 2, SlimPer achieves $\mathcal{O}(\mathbf{L} \cdot \mathbf{N} \cdot \mathbf{K})$ user–item interaction capacity—matching DLRM and $\mathbf{K} \times$ larger than transformer–like models that achieve only $\mathcal{O}(\mathbf{L} \cdot \mathbf{N})$. The difference arises because transformers allocate the majority of their $\mathcal{O}(\mathbf{N}^2)$ computation to pairwise user–side token interactions that do not directly serve relevance estimation, whereas SlimPer directs all $\mathbf{K} \times \mathbf{N}$ interactions per layer toward knowledge–base–to–token relevance matching. The bottleneck thus constrains *intermediate memory*, not *learning capacity for user–item relevance*. Notably, this capacity argument is independent of whether attention is quadratic or linearized. Sub–quadratic variants such as linear attention (Katharopoulos et al., 2020) and sliding–window attention (Beltagy et al., 2020) reduce per–layer compute to $\mathcal{O}(\mathbf{N})$, but they still (i) propagate $\mathbf{N}$ –sized token–level intermediate tensors across depth, leaving memory at $\mathcal{O}(\mathbf{L} \cdot \mathbf{N})$ rather than $\mathcal{O}(\mathbf{L} \cdot \mathbf{K})$, and (ii) devote their interaction capacity to user–side token mixing, leaving user–item interaction capacity at $\mathcal{O}(\mathbf{L} \cdot \mathbf{N})$ —a factor of $\mathbf{K}$ below SlimPer. We therefore expect SlimPer to retain a quality advantage over efficient–attention baselines, not merely an efficiency one, because the gain stems from *how* capacity is allocated rather than from lowering the asymptotic order of attention.

Iterative access prevents irreversible loss. Standard bottleneck architectures are constrained by the Data Processing Inequality: for a Markov chain $\mathbf{X} \rightarrow \mathbf{T} \rightarrow \mathbf{Y}$, information discarded during compression is irrecoverable. SlimPer breaks this chain—each layer cross–attends directly into the raw input tokens $\mathbf{X}$, not only the compressed representation from the previous layer. In the language of the Information Bottleneck framework (Tishby et al., 2000), SlimPer performs *iterative* compression: each layer extracts additional task–relevant mutual information $I(\mathbf{x}^k; \mathbf{Y})$ while the raw evidence remains available to correct earlier omissions—a property validated by the attention heatmaps in Appendix E, which show that successive layers progressively narrow their focus from broad historical context to the most relevant recent signals.

3. Related Work

We situate SlimPer within three lines of research: feature interaction learning for fixed-size inputs, user history modeling for variable-length event sequences, and cross-attention bottleneck architectures from adjacent domains.

3.1. Feature Interaction Learning

This paradigm focuses on capturing complex, high-order interactions among fixed-size input features. Factorization Machines (Rendle, [2010](#)) and DeepFM (Guo et al., [2017](#)) introduced efficient second-order interaction modeling through pairwise factorization. DLRM (Naumov et al., [2019](#)) combined sparse embeddings with MLP-processed dense features and explicit dot-product interactions, a design that has since been widely cited. Deep Cross Network (DCN) (Wang et al., [2017](#)) and DCNv2 (Wang et al., [2021](#)) proposed cross networks to model bounded-degree feature crosses, while AutoInt (Song et al., [2019](#)) applied multi-head self-attention over feature embeddings to learn implicit higher-order interactions. More recent work emphasizes architectural scalability: DHEN (Zhang et al., [2022](#)) and Wukong (Zhang et al., [2024](#)) systematically studied stacking interaction modules, demonstrating favorable scaling laws for fixed-size features.

Relation to SlimPer. These approaches treat user history as a pre-aggregated feature vector. SlimPer subsumes their capability through its knowledge base refinement while additionally enabling direct access to individual user history events at every layer.

3.2. User History Modeling

Effectively capturing signals from sequence features has become increasingly important as user history data grows richer. DIN (Zhou et al., [2018](#)) and DIEN (Zhou et al., [2019](#)) introduced target-aware attention over user interaction histories for intent modeling. Transformer-based models (BST (Chen et al., [2019](#)), BERT4Rec (Sun et al., [2019](#)), SASRec (Kang and McAuley, [2018](#))) treat user events as tokens and apply full self-attention, achieving expressive user representations but incurring $\mathcal{O}(N^2)$ cost that limits scaling.

Research on ultra-long user histories has explored several strategies: (1) *Retrieval-based*: SIM (Pi et al., [2020](#)) uses a two-stage hard search to retrieve relevant behaviors; (2) *Memory-based*: MIMN (Pi et al., [2019](#)) maintains fixed-size representations of long-term interests; (3) *Compression-based*: LONGER (Chai et al., [2025](#)) applies token merging, and HSTU (Zhai et al., [2024](#)) uses stochastic length sampling; (4) *Offline aggregation*: TWIN-V2 (Si et al., [2024](#)) and DV365 (Lyu et al., [2025](#)) shift long-range modeling to pre-computed embeddings. A common pattern across these strategies is that they reduce the cost of

long histories by limiting how much raw behavioral detail is available for modeling user–item interactions, trading fine–grained, target–aware access to user history information for tractability at scale.

Despite these advances, user history modeling typically remains a separate component whose output is summarized and combined with fixed–size features via late fusion, limiting deep cross–modality interactions. Recent unified architectures, such as OneTrans (Zhang et al., [2025](#)), HHFT (Yu et al., [2025](#)), RankMixer (Zhu et al., [2025](#)), process all modalities jointly using Transformer–style designs, but inherit the same quadratic cost when scaling to long sequences.

Relation to SlimPer. SlimPer resolves this trade–off with $\mathcal{O}(K \cdot N)$ per–layer compute and a fixed–size memory footprint that does not grow with sequence length, scaling to 10k+ events while accessing raw user–side tokens at every layer without information loss.

3.3. Cross–Attention Bottleneck Architectures

SlimPer’s use of a fixed–size knowledge base that iteratively cross–attends into a larger token set shares structural resemblance with architectures from other domains. Perceiver (Jaegle et al., [2021](#)) and Perceiver IO (Jaegle et al., [2022](#)) maintain a learned latent array that iteratively cross–attends into raw inputs, achieving linear complexity in input length. Set Transformer (Lee et al., [2019](#)) introduces Induced Set Attention Blocks (ISAB) with a similar bottleneck for set–structured data. Flamingo (Alayrac et al., [2022](#)) applies cross–attention from a language model into visual tokens for multi–modal understanding.

While SlimPer shares the principle of decoupling computation from input length through a compact bottleneck, it introduces several recommendation–specific inductive biases that are absent from general–purpose architectures: (1) **discriminative relevance focus**: the knowledge base is initialized from item–side features and dedicated entirely to $user, item$ relevance scoring; (2) **explicit relevance matching**: a dedicated dot–product step (Step 2) computes scores between retrieved evidence and multifaceted templates, with no analog in Perceiver–style designs; (3) **ROO awareness**: user–side tokens are computed once and shared across all candidates within a request, reducing redundant computation and memory usage; and (4) **modality–dependent attention kernels**: SlimPer employs parametric MLP–based attention for fixed–size sparse tokens and scaled dot–product attention for variable–length sequence features.

4. Experiments

4.1. Experiment Setup

Dataset

We conduct experiments on Instagram’s final-stage ranking models using the same training and evaluation data used to train the current models.

Evaluation Metrics

We evaluate model performance using Normalized Entropy (NE), a standard offline metric that exhibits directional alignment with A/B test results. NE normalizes the model’s binary cross-entropy loss (BCEloss) by dividing it by the BCEloss of a naive baseline that predicts the global label average p_k for all samples. An NE improvement of approximately 0.03% is considered empirically significant.

$$(12) \quad \text{NE}_k = \frac{\text{BCEloss}(\hat{y}^{(k)}, y^{(k)})}{-\left[p_k \ln(p_k) + (1 - p_k) \ln(1 - p_k) \right]},$$

Baseline

We compare SlimPer against Instagram’s existing models, which employ a late-fusion architecture that stitches together HSTU (Zhai et al., [2024](#)) for modeling variable-length user interaction history sequences and Wukong (Zhang et al., [2024](#)) for modeling fixed-size sparse/dense features. These models perform the final-stage ranking for Instagram’s major surfaces and represent an established baseline. To ensure a fair comparison, we report NE improvements relative to the baseline using the same training and evaluation date ranges. All other experimental settings are held constant across models, including optimizer configuration and feature sets, unless otherwise noted.

4.2. Offline Performance

We conducted an offline evaluation of SlimPer against the baseline on both Reels and Feed. The main results are summarized in Table [3](#). We make three observations that are consistent across both surfaces: (1) SlimPer achieves consistent NE improvements across all tasks. Due to space constraints, we report only representative tasks for both surfaces. (2) At the same sequence length, SlimPer delivers both NE wins and QPS wins compared to the baseline, in-

dicating that the proposed design improves prediction quality while also improving efficiency. (3) We observe a scaling trend: NE wins increase as sequence length grows. For example, on Feed reshare prediction, NE improvements grow from -0.49% at 1k sequence length to -0.94% at 4k sequence length, corresponding to close to $2\times$ relative gain in model quality, albeit with a 16% QPS regression.

These empirical findings are consistent with the Big-O analysis in Table 2: SlimPer’s linear-in- N computational complexity and its $\mathcal{O}(L \cdot K \cdot d)$ memory footprint directly account for the observed 11%+ QPS improvement and 9–18% memory reduction compared to the quadratic-cost baseline. Because this cost reduction stems from replacing self-attention over the full sequence with fixed-size knowledge base refinement, it generalizes to any transformer-like method that processes the full token sequence via quadratic self-attention.

Table 3. SlimPer vs. baseline. NE and QPS are relative changes (%); Memory is absolute change in average usage (%); all w.r.t. surface-specific baselines. Lower NE/Memory and higher QPS are better.

Surface	Model Configuration	NE (%)						QPS (%)	Memory Diff (%)
		reshare	skip	like	comment	follow	save		
Reels	Baseline w/ 2k events	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SlimPer w/ 2k events	-0.51	-0.31	-0.23	-0.41	-0.35	-0.35	+11.0	-9.32
	SlimPer w/ 5k events	-0.80	-0.57	-0.49	-0.64	-0.53	-0.62	-10.0	+4.59
Feed	Baseline w/ 1k events	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	SlimPer w/ 1k events	-0.49	-0.28	-0.54	-0.52	-0.71	-0.62	+12.5	-18.12
	SlimPer w/ 4k events	-0.94	-0.52	-0.91	-0.94	-1.00	-1.05	-16.07	+2.04

4.3. A/B Test Performance

We launched SlimPer w/ **5k events** for Instagram Reels and SlimPer w/ **4k events** for Instagram Feed, the models described in Section 4.2. After pre-testing and backtesting,

SlimPer was launched to full traffic on both surfaces, where it delivers statistically significant gains across multiple major engagement metrics, with an aggregate topline impact roughly $10\times$ that of a typical statistically significant launch—a notable improvement from modeling changes in 2025. Moreover, SlimPer introduces no regression on ecosystem guardrail metrics (e.g., integrity, diversity, and recency) and maintains training and inference GPU capacity at approximately neutral levels, confirming that the efficiency gains observed offline (Section [4.2](#)) translate to real-world serving.

In 2026, we have launched additional SlimPer variants that scale to 10k+ events with increased model capacity. These launches have delivered major engagement wins across multiple product surfaces at Meta, further validating SlimPer’s efficacy and scalability under real-world constraints.

4.4. Scaling and Efficiency Analysis

The Big-O analysis in Table [2](#) predicts that SlimPer’s $\mathcal{O}(K \cdot N)$ per-layer complexity should yield clear efficiency advantages over quadratic baselines, with the gap widening as sequence length grows. We validate this prediction empirically by comparing SlimPer against the baseline (late-fusion HSTU + Wukong) across different sequence length configurations, using the Reels model as a representative example. Whereas the baseline’s usable history length is constrained by the quadratic attention cost over the sequence, SlimPer avoids this bottleneck. Results are summarized in Figure [4](#).

Main findings. At matched 2k length, SlimPer outperforms the baseline by -0.51% NE (re-share) while also delivering higher throughput ($+11\%$ QPS) and lower memory—improving quality and efficiency simultaneously. The gain compounds with sequence length: at 6k, SlimPer reaches -0.86% NE at under 20% QPS regression, whereas the baseline gains only -0.13% NE while losing 48% QPS. As the curves show, SlimPer’s NE improves monotonically as the compact knowledge base absorbs richer long-term signals, while the quadratic-cost baseline sees diminishing returns at longer sequences.

Scaling and Efficiency Analysis

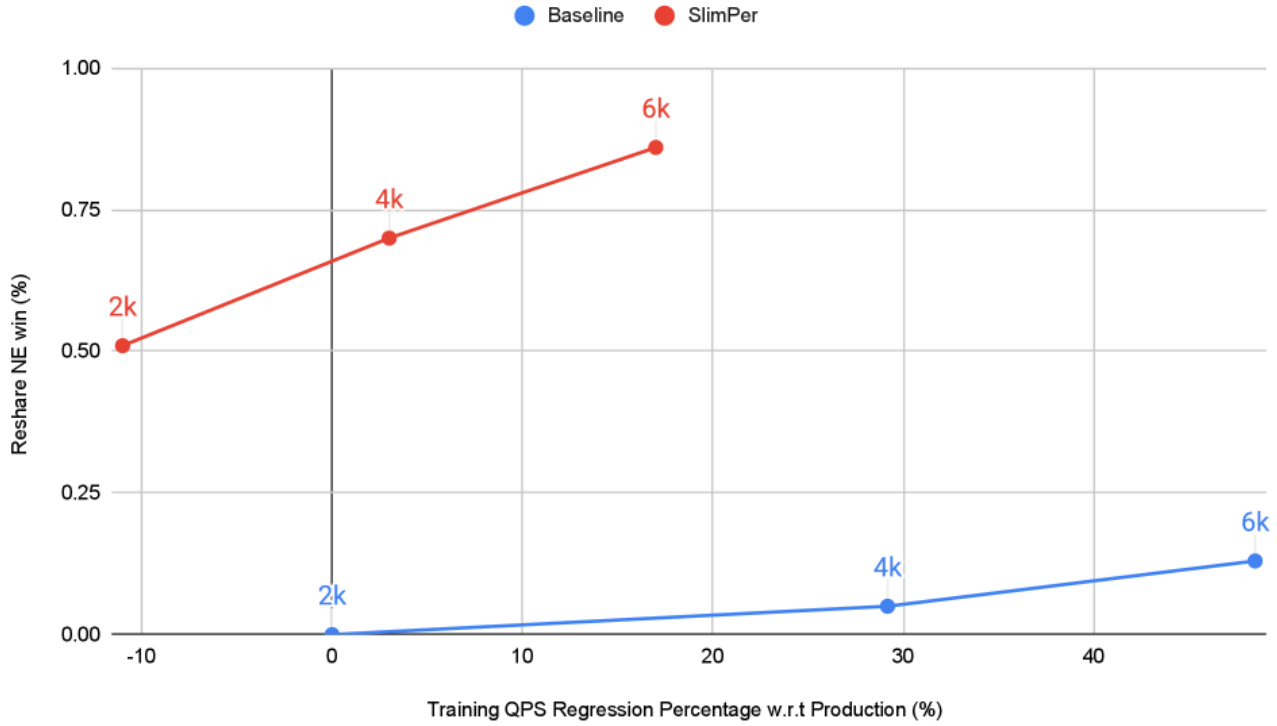


Figure 4. Quality—efficiency scaling as user—history sequence length grows (2k—6k). Each curve traces one model (SlimPer vs. the baseline) across sequence lengths; the x -axis is training QPS regression and the y -axis is NE win, both relative to the baseline (%). SlimPer sustains larger NE wins at smaller QPS regressions, and its advantage widens as sequence length increases.

These results confirm that SlimPer’s quality gains come not from spending more compute, but from allocating it more effectively toward user—item relevance matching. Consistent with Table 2 and the information—theoretic analysis in Section 2.7, SlimPer’s $\mathcal{O}(K \cdot N)$ per-layer cost translates to 8—25 $\times$ fewer FLOPs in practice (Appendix E), with the advantage growing superlinearly in input length: the compact knowledge base redirects interaction capacity away from the $\mathcal{O}(N^2)$ user—side—to—user—side computations that dominate transformer—like models and toward $\mathcal{O}(K \times N)$ relevance—focused interactions per layer.

Memory efficiency.

The observed 9—18% memory reduction at matched sequence length stems from two complementary mechanisms: the fixed—size knowledge base replaces $\mathcal{O}(N)$ —sized intermediate

tensors at every layer, and the ROO-aware design shares a single copy of all user-side tokens across candidates within each request.

4.5. Ablation Studies

We conduct ablation experiments to validate our design decisions.

4.5.1. Effect of Contextual Encoding

We ablate whether incorporating surrounding event information into each history token improves performance by comparing three strategies: (1) no context (independent encoding), (2) event-type context (encoding surrounding actions, e.g., like, reshare, long watch), and (3) full context (event-type + event content jointly). Introducing contextual encoding yields approximately **−0.15% NE** improvement on the reshare task. Notably, the gain comes primarily from encoding event type alone; adding event content provides only marginal additional benefit (**< 0.02% NE**). This suggests that users’ behavioral patterns—*what actions* they take—are more informative than the specific items they interact with, highlighting an important difference from language modeling where content dependencies determine meaning.

4.5.2. Effect of Knowledge Base Size and Model Depth

We further ablate the knowledge base size K and the number of stacked SlimPer layers. A knowledge base size of $K=64$ provides a favorable quality–efficiency balance; reducing to $K=4$ yields +23.2% QPS but degrades NE by +1.2%, confirming the knowledge base’s role as an effective information bottleneck. For model depth, 7 layers offers strong performance; gains continue up to 9 layers (**−0.12% NE**) with moderate cost increase. Full ablation tables are provided in Appendix [D](#).

4.6. Interpretability of Model Predictions

SlimPer’s attention mechanism enables direct attribution of recommendations to specific user history events. For a given request, the model produces attention scores between candidate items and sequence tokens, allowing practitioners to identify the top- k historical interactions that most influenced each prediction. Spot-check analyses confirm that attention weights align with ranking outputs, indicating high matching fidelity. We provide a detailed analysis of attention patterns across model layers in Appendix [E](#).

For reproducibility details including hyperparameter settings, see Appendix [A](#).

5. Limitations

Limited direct comparison with recent unified architectures.

We compare SlimPer against a mature baseline (HSTU + Wukong) rather than directly against recent transformer-style unified architectures (e.g., OneTrans, HHFT, RankMixer). In practice, a standalone unified HSTU performs worse than this hybrid, which uses dedicated components per modality (HSTU for user history, Wukong for fixed-size features); the hybrid is therefore a stronger comparison point than any single unified architecture, and SlimPer outperforms it by a clear margin while also simplifying the system into a single backbone. Differences in training data, feature sets, and serving constraints across organizations further make reported numbers from other systems non-comparable. Our complexity analysis (Table 2) is nonetheless general: any architecture using quadratic self-attention over the full sequence faces the same $\mathcal{O}(N^2)$ compute bottleneck, and any architecture maintaining N -sized intermediate tensors or duplicating user-side representations per candidate faces the same memory bottleneck—both of which SlimPer’s fixed-size, ROO-aware knowledge base resolves.

Reproducibility on internal data.

All experiments are conducted on internal Instagram training and evaluation data, which limits independent reproduction of our exact results. Furthermore, our models are trained to convergence on data volumes not readily available in public benchmarks, which further limits exact reproducibility but also validates SlimPer’s effectiveness in deployment. To mitigate this, we provide detailed architectural hyperparameters (Appendix A), describe the model using only standard building blocks (MLPs, QKV attention, linear projections, RMSNorm), and report relative improvements against a clearly defined baseline under controlled experimental conditions. We believe these details are sufficient for practitioners to reimplement and validate the approach on their own recommendation datasets.

6. Conclusion

We presented SlimPer, a unified personalization model that reframes recommendation ranking as iterative refinement of a compact $\langle user, item \rangle$ knowledge base. By departing from the transformer-mimicking paradigm dominant in recent work, SlimPer achieves a distinct efficiency-quality trade-off: model depth is decoupled from input sequence length, raw multi-modal evidence is preserved uncompressed and accessed directly at every layer, and the entire model capacity is focused on relevance estimation rather than maintaining token-level representations. SlimPer further minimizes GPU memory by eliminating N -sized intermediate

tensors across layers and maintaining a single shared copy of user-side tokens across all candidates within a request via request-only optimization.

Our experiments on an industrial recommender system validate the effectiveness of SlimPer. The proposed architecture achieves strong offline performance with lower computational cost than transformer-based alternatives, and delivers statistically significant engagement improvements in controlled A/B experiments. Moreover, SlimPer is inherently interpretable: attention weights at each refinement stage reveal how evidence is selected and aggregated, facilitating principled debugging, auditing, and model analysis.

Looking ahead, we see several promising directions for future work. First, the iterative refinement paradigm naturally extends beyond sequential user histories to evidence sources without inherent ordering, such as cross-surface engagement signals or candidates generated by upstream selection systems. Because the knowledge base interacts directly with raw tokens rather than relying on sequence structure, SlimPer can integrate such signals without architectural changes. Second, SlimPer’s whole-model ROO-aware design enables efficient deployment across ranking stages, such as early-stage ranking, where larger candidate sets further increase the benefits of amortized computation. Finally, the current matching and refinement modules are intentionally lightweight, relying on simple dot-product matching and MLP-based updates. Exploring more expressive operators within these components may yield additional quality improvements while preserving the iterative-refinement framework with its $\mathcal{O}(N)$ per-layer complexity and fixed-size memory footprint. More broadly, we believe that recommendation-centric architectural design around the unique computational and modeling requirements of personalization will continue to drive practical advances in industrial recommendation systems.

Acknowledgements.

Engineering Collaborators

We are grateful to the many collaborators across Meta whose partnership has been invaluable in building and scaling SlimPer across our family of apps: Jay Wang, Keming Zhang, Chen Chen, Qichao Que, Wanli Ma, Maxwell Lin-He, Yu Liu, Yanhong Wu, Xinye Zheng, Wen-Yun Yang, Rui Zhang, Jiyan Yang, Zichao Li, Lei Nie, Yan Sun, Rongcheng Lin, Ivan Ji, Cory Nezin, Jeffrey Cordero, Lambert Chu, Leon Yan, Keyi Wu, Yiming Liao, Irra Na, Neel Pawar, Tingting Zhang, Xue Zou, Ziwei Li, Zhe (Joe) Wang, Haifeng Lan, Qinjie Lin, Ye Wang, Huasha Zhao, Jianming He, William Pei, Dawei Fan, Yezhou Huang, Minhan Li, Zhiwei

Zhang, Dev (Devashish) Shankar, Jackie (Jiaqi) Xu, Chunzhi Yang, Haoxun Luo, Hao Lin, Prashasti Baid, Lu Zhang, Liang Guo, and Rui Jian.

Leadership

We are also grateful for the leadership support spanning Meta’s family of apps and MRS: Jayant Subramanian, Haotian Wu, Jing Qian, Hui Zhang, Shilin Ding, Sophia (Xueyao) Liang, Xinyao Hu, Linhong Zhu, Yisong Song, Shujian Bu, Aameek Singh, Hong Yan, Meihong Wang, Mahesh Srinivasan, Adam (Yang) Song, Nipun Mathur, Lars Backstrom, Jeff Huang, Tessa Lyons–Laing, Deepak Agarwal, and Max Eulenstein, whose guidance and investment made this work possible.

Appendix A Reproducibility

Table 4 summarizes the main architectural hyperparameters for SlimPer on Feed and Reels. All models are trained for one epoch with the Shampoo optimizer (Gupta et al., 2018) at a learning rate of 0.05, with identical optimizer settings across all model configurations (both SlimPer and baseline).

Table 4. SlimPer architectural hyperparameters for Feed and Reels.

Hyperparameter	Feed	Reels
Number of SlimPer layers (L)	5	7
Knowledge base size (K)	64	64
Hidden dimension (d)	256	256
Query slots (q)	16	16
Template slots (t)	32	32

Appendix B Notation and Preliminaries

To reduce architectural complexity and simplify the implementation, we restrict the model to a minimal set of building blocks with identical internal structures, differing only in hyperpara-

meter settings. We define the following notations and operator blocks used throughout the network.

Multilayer Perceptrons (MLP)

MLP denotes the standard multilayer perceptron, a stack of fully connected layers with non-linear activations applied between layers. In our experiments, we use SiLU as the activation function.

Segment-wise Linear Projection ($\mathcal{L}$)

$\mathcal{L}$ is a segment-wise linear projection module that transforms an input tensor $\alpha \in \mathbb{R}^{L_{\text{in}} \times d}$ into an output tensor $\beta \in \mathbb{R}^{L_{\text{out}} \times d}$ using a learnable projection matrix ρ :

$$(13) \quad \beta = \mathcal{L}(\alpha) = \rho\alpha, \quad \rho \in \mathbb{R}^{L_{\text{out}} \times L_{\text{in}}}.$$

In our network, we use this module multiple times, each with independent learnable parameters. For simplicity, we omit the distinguishing subscripts and superscripts in subsequent equations.

All-Modality-Aware $\langle user, item \rangle$ Knowledge Base ($\mathcal{X}$)

We use $\mathcal{X}$ to denote the all-modality $\langle user, item \rangle$ knowledge base. This knowledge base is initialized at Layer-0 as:

$$(14) \quad \mathcal{X}^0 = \mathcal{L}(\mathbf{S}_{\text{in}})$$

where $\mathbf{S}_{\text{in}}$ represents the non-user-side input tokens, typically derived from item-side or cross-side sparse features. The shape of $\mathcal{X}^k$ can vary across layers, but in our experiments, we use a consistent shape of $K \times d$ (with $K = 64$) for all layers.

Appendix C Tokenization Details

All raw input features are first converted into a set of modality-specific tokens within the unified backbone. This step preserves the semantic distinctions of the feature types while

preparing them for the shared interaction stack. We consider three types of input modalities: sparse features, user history sequence features and dense features. The corresponding tokenization approaches are summarized in Table 5.

Table 5. Continuous tokenization approaches across different information modalities.

Modality	Continuous Tokenization Approaches
Sparse	Embedding Pooling (E-P): Sparse features are tokenized via sum or mean pooling over embeddings, yielding one or a small number of fixed-size vectors (e.g., $1 \times d$).
Sequence	Event Modeling (E-M): All information describing each event is aggregated and fused into a fixed-length event representation (e.g., $1 \times d$ per event).
Dense	Dense Processing (D-P): Dense features are normalized and projected by an MLP into a fixed-size vector (e.g., 1024).

C.0.1. Multi-modality Tokenization

Embedding-Pooling Module (E-P)

Sparse features are first mapped to embedding vectors through standard embedding lookup tables. For features with multiple values (e.g., a user’s top- k liked categories), we apply pooling operations:

$$(15) \quad \mathbf{S}^j = \text{Pool}(\{\mathbf{e}_{j,1}, \mathbf{e}_{j,2}, \dots, \mathbf{e}_{j,k}\})$$

where $\text{Pool}(\cdot)$ can be mean or sum pooling. Concatenating the pooled results from all sparse features, we obtain the tensor representing sparse feature tokens as $\mathbf{S} = (\mathbf{S}^{(1)}, \dots, \mathbf{S}^{(|S|)}) \in \mathbb{R}^{|S| \times d}$, with d as the dimension for each token.

Event-Modeling Module (E-M)

The E-M module processes each event in the user history into a token representation. Each event is typically described by several sparse features representing its content, the user’s explicit action types, implicit watch time, engagement timestamps, and other metadata. We use

$\mathbf{E} = (\mathbf{E}^{(1)}, \dots, \mathbf{E}^{(t)}, \dots)$ to represent the tokenized user history event sequence. To obtain each $\mathbf{E}^{(t)}$, the E–M module builds several parallel encoding components, projects each to the shared embedding dimension d , and combines them via summation:

$$(16) \quad \mathbf{E}^{(t)} = \text{MLP}_E^s(\mathbf{e}_{\text{sparse}}^{(t)}) + \mathbf{e}_{\text{temporal}}^{(t)} \\ + \text{MLP}_E^e(\mathbf{e}_{\text{type}}^{(t)}) + \mathbf{e}_{\text{context}}^{(t)}$$

where $\mathbf{e}_{\text{sparse}}^{(t)}$ is the content embedding from concatenated pooled sparse features describing the event item, $\mathbf{e}_{\text{type}}^{(t)}$ is the event type encoding that captures both explicit and implicit actions, and $\mathbf{e}_{\text{temporal}}^{(t)}$ is the temporal encoding, which is the sum of positional encoding and delta–time encoding.

Dense–Processing Module (D–P)

Dense features are concatenated and projected into a fixed–size vector through a simple MLP:

$$(17) \quad \mathbf{D} = \text{MLP}\left(\text{Concat}(\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_{|\mathcal{D}|})\right)$$

Appendix D Ablation Studies

We conduct ablation experiments to validate our design decisions underlying SlimPer’s iterative refinement paradigm. The baseline corresponds to the **Reels SlimPer model with 5k events** used in Table 3 with knowledge base size $K = 64$ and 7 refinement layers.

D.0.1. Effect of Knowledge Base Size

Table 6 presents an ablation study on the number of knowledge base slots K (i.e., $K \in \{4, 8, 16, 32, 64\}$), which directly evaluates the information capacity of the iterative refinement bottleneck. The baseline configuration has $K = 64$. For each configuration, the query size q is defined as one quarter of the knowledge base size, while the template size t is defined as one half of K .

Table 6. Ablation study on knowledge base size (relative to the $K = 64$ baseline).

K, q, t	NE (%)	QPS (%)	Mem Diff (%)
(64, 16, 32)	0.0	0.0	0.0
(4, 1, 2)	+1.2	+23.2	−9.53
(8, 2, 4)	+0.53	+21.2	−8.19
(16, 4, 8)	+0.36	+17.8	−7.43
(32, 8, 16)	+0.18	+4.25	−1.31

D.0.2. Effect of Number of Refinement Layers

Table 7 presents an ablation study on the number of stacked SlimPer refinement layers (i.e., $L \in \{3, 5, 7, 9\}$). This experiment evaluates whether iterative refinement consistently improves ranking quality and examines the depth at which performance gains begin to saturate. The baseline configuration has 7 refinement layers.

Table 7. Ablation study on the number of SlimPer refinement layers (relative to the 7-layer baseline).

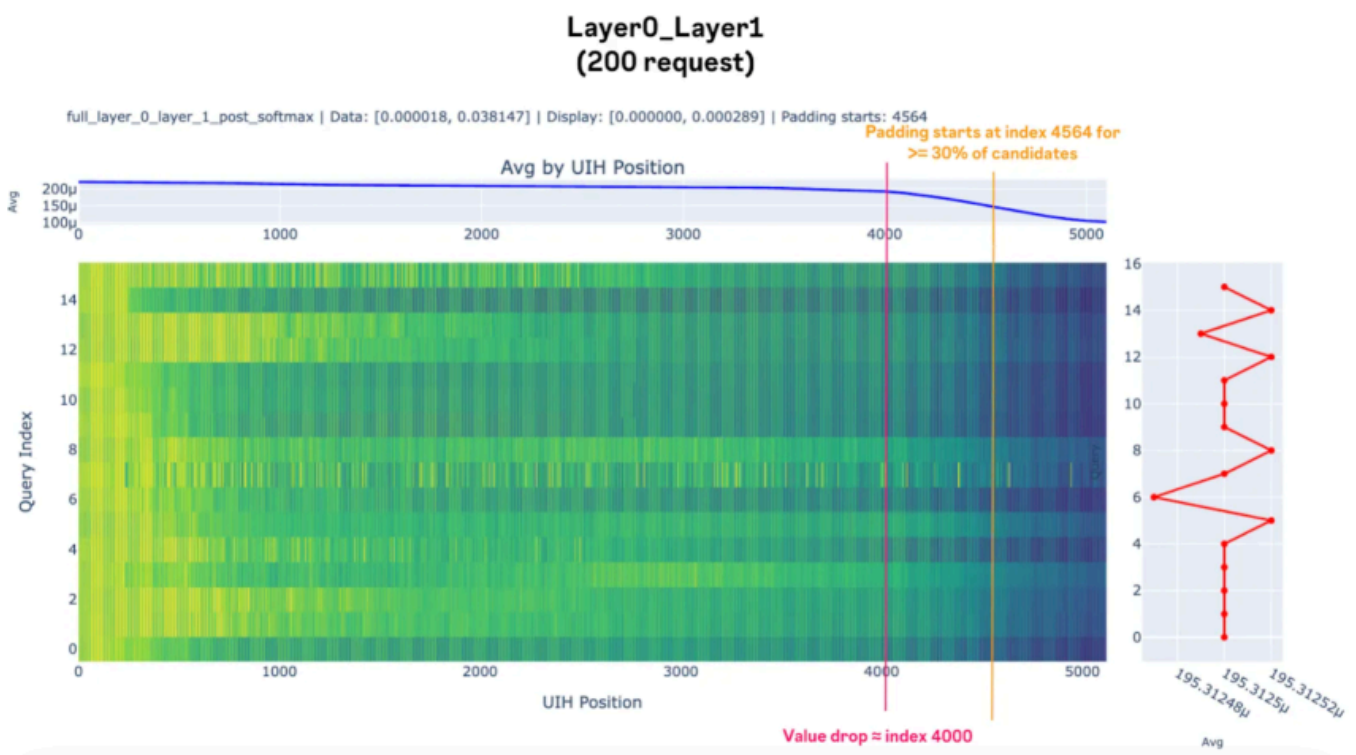
Layer	NE (%)	QPS (%)	Mem Diff (%)
7	0.0	0.0	0.0
3	+0.32	+21	−8.60
5	+0.14	+8.8	−5.50
9	−0.12	−7.16	+4.92

Appendix E Interpretability of Model Predictions

SlimPer’s iterative refinement design provides intrinsic interpretability: the QKV attention at each layer directly links past user interactions to the current prediction, making it possible to attribute recommendations to specific historical events. Unlike prior models that stitch together opaque components, SlimPer’s attention scores offer a natural explanation for why an item is recommended to a user.

Attention Analysis Across Layers.

For a given request, the model produces an attention matrix mapping candidate items (queries) to sequence tokens (keys). We aggregate attention distributions across 200 requests with 5k sequence length and visualize them as heatmaps, where the x-axis represents sequence indices in sequential position and the y-axis represents candidate items.



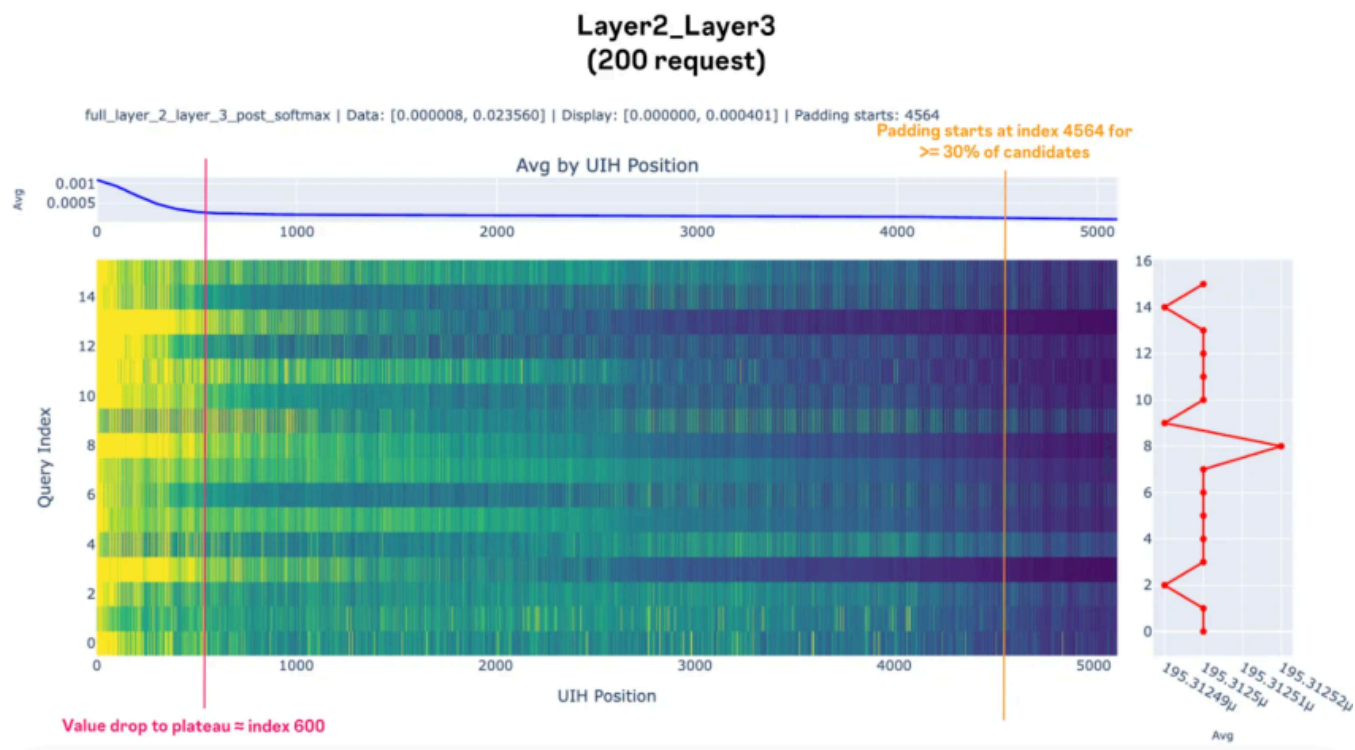


Figure 5. Attention heatmaps for lower and middle SlimPer layers. Lower layers attend broadly across the full history; middle layers concentrate on more recent events.

As shown in Fig. 5, at the lowest layers the model attends broadly across the first 4k sequence indices, capturing coarse-grained historical context. At the middle layers, attention concentrates on the first few hundred indices, focusing on a narrower and more recent subset of interactions.

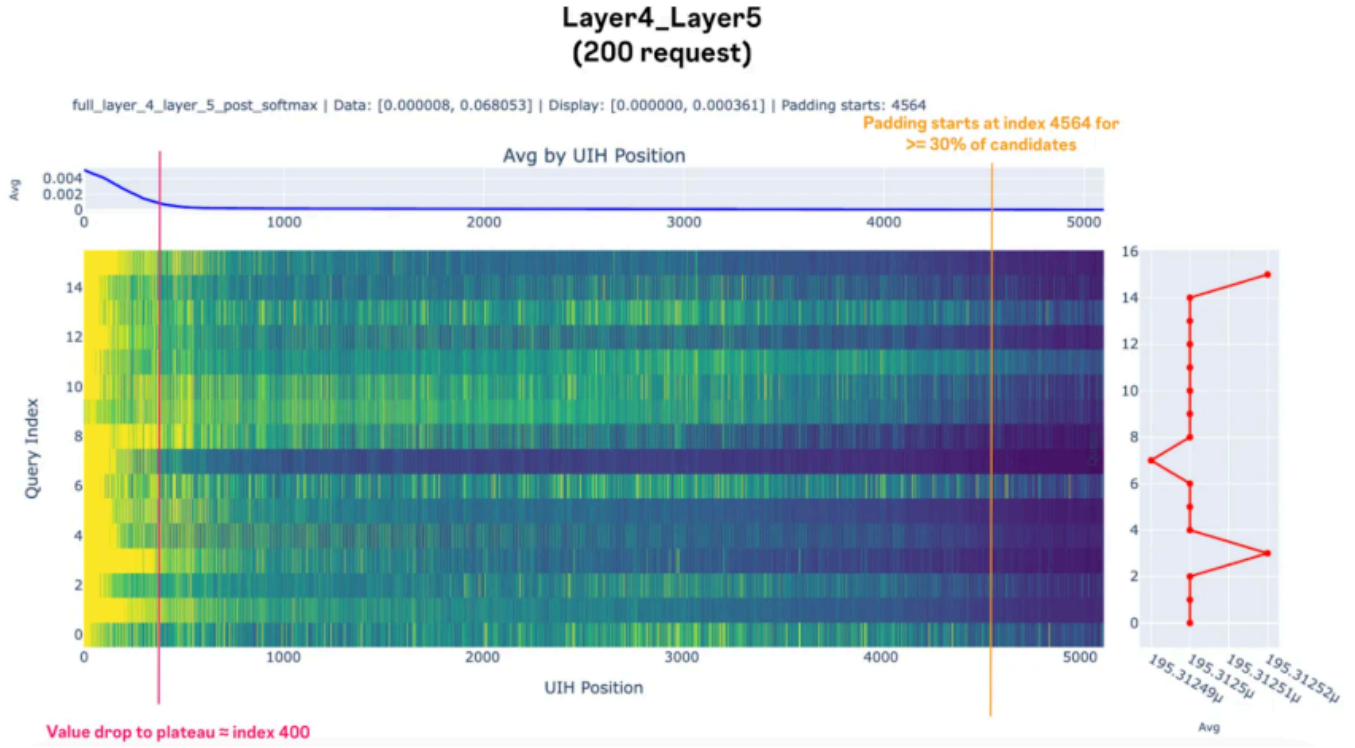


Figure 6. Attention heatmap for upper SlimPer layers. Attention further narrows to the most recent ~ 400 events.

At the upper layers (Fig. 6), the pattern sharpens further—attention is concentrated within the most recent ~ 400 events. This hierarchical attention behavior—broad context at lower layers, progressively narrower focus at upper layers—is consistent with SlimPer’s iterative refinement design: earlier layers gather broad evidence about user preferences, while later layers refine predictions using the most recent and most influential behavioral signals.

Appendix F FLOPs Analysis

Beyond throughput (QPS), we quantify the computational cost by counting all matrix multiplications in the forward pass (excluding embedding lookups, normalizations, and activations). Table 8 reports the estimated FLOPs at different sequence lengths. At the shared default of 2k, SlimPer uses $\sim 8\times$ fewer FLOPs than the baseline. The gap widens with sequence length: at 6k, SlimPer (~ 6 GFLOPs) still uses $\sim 4\times$ fewer FLOPs than the baseline at its native 2k (~ 24 GFLOPs), while achieving -0.86% NE improvement.

Table 8. Estimated forward-pass FLOPs per example (GFLOPs), counting all matrix multiplications. Baseline counts the HSTU encoder (the sequence-length-dependent component) only.

Seq. Len.	Baseline (HSTU)	SlimPer	Ratio
2048	~ 24	~ 3	~8 ×
4096	~ 74	~ 5	~16 ×
6144	~ 150	~ 6	~25 ×

References

J. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan (2022)

I. Beltagy, M. E. Peters, and A. Cohan (2020)

Z. Chai, Q. Ren, X. Xiao, H. Yang, B. Han, S. Zhang, D. Chen, H. Lu, W. Zhao, L. Yu, X. Xie, S. Ren, X. Sun, Y. Tan, P. Xu, Y. Zheng, and D. Wu (2025)

Q. Chen, H. Zhao, W. Li, P. Huang, and W. Ou (2019)

Flamingo: a visual language model for few-shot learning.

In Advances in Neural Information Processing Systems (NeurIPS),
Cited by: [§3.3](#).

Longformer: the long-document transformer.
arXiv preprint arXiv:2004.05150.
Cited by: [§2.5](#), [§2.7.1](#).

LONGER: scaling up long sequence modeling in industrial recommenders.
External Links: 2505.04421, [Link](#)
Cited by: [§3.2](#).

Behavior sequence transformer for e-commerce recommendation in alibaba.
In Proceedings of the 1st international workshop on deep learning practice for high-dimensional sparse data, pp. 1–4.
Cited by: [§3.2](#).

H. Guo, R. Tang, Y. Ye, Z. Li, and X. He (2017)

DeepFM: a factorization-machine based neural network for ctr prediction.

In Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI),

pp. 1725–1731.

Cited by: [§3.1](#).

V. Gupta, T. Koren, and Y. Singer (2018)

Shampoo: preconditioned stochastic tensor optimization.

In International Conference on Machine Learning,

pp. 1842–1850.

Cited by: [Appendix A](#).

A. Jaegle, S. Borgeaud, J. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, O. Hénaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira (2022)

Perceiver IO: a general architecture for structured inputs & outputs.

In International Conference on Learning Representations (ICLR),

Cited by: [§3.3](#).

A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira (2021)

Perceiver: general perception with iterative attention.

In International Conference on Machine Learning (ICML),

Cited by: [§3.3](#).

W. Kang and J. McAuley (2018)

Self-attentive sequential recommendation.

In 2018 IEEE international conference on data mining (ICDM),

pp. 197–206.

Cited by: [§3.2](#).

A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret (2020)

Transformers are RNNs: fast autoregressive transformers with linear attention.

In ICML,

pp. 5156–5165.

Cited by: [§2.7.1](#).

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner (1998)

Gradient-based learning applied to document recognition.
Proceedings of the IEEE **86** (11),
pp. 2278–2324.
Cited by: [§2.5](#).

J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh (2019)

Set transformer: a framework for attention-based permutation-invariant neural networks.
In International Conference on Machine Learning (ICML),
Cited by: [§3.3](#).

W. Lyu, D. Tyagi, Y. Yang, Z. Li, A. Somani, K. Shanmugasundaram, N. Andrejevic, F. Adepotra, C. Zeng, A. K. Singh, M. Ransan, and S. Jain (2025)

DV365: extremely long user history modeling at instagram.
In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '25),
pp. 4717–4727.
Cited by: [§3.2](#).

M. Naumov, D. Mudigere, H. M. Shi, J. Huang, N. Sundaraman, J. Park, X. Wang, U. Gupta, C. Wu, A. G. Azzolini, et al. (2019)

Deep learning recommendation model for personalization and recommendation systems.
arXiv preprint arXiv:1906.00091.
Cited by: [§3.1](#).

Q. Pi, W. Bian, G. Zhou, X. Zhu, and K. Gai (2019)

Practice on long sequential user behavior modeling for click-through rate prediction.
In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining,
pp. 2671–2679.
Cited by: [§3.2](#).

Q. Pi, G. Zhou, Y. Zhang, Z. Wang, L. Ren, Y. Fan, X. Zhu, and K. Gai (2020)

Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction.

In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2685–2692.

Cited by: [§3.2](#).

S. Rendle (2010)

Factorization machines.

In 2010 IEEE International Conference on Data Mining, Vol. , pp. 995–1000.

Cited by: [§3.1](#).

Z. Si, L. Guan, Z. Sun, X. Zang, J. Lu, Y. Hui, X. Cao, Z. Yang, Y. Zheng, D. Leng, et al. (2024)

Twin v2: scaling ultra-long user behavior sequence modeling for enhanced ctr prediction at kuaishou.

In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, pp. 4890–4897.

Cited by: [§3.2](#).

W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang (2019)

AutoInt: automatic feature interaction learning via self-attentive neural networks.

In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19), pp. 1161–1170.

Cited by: [§3.1](#).

F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang (2019)

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.

Cited by: [§3.2](#).

N. Tishby, F. C. Pereira, and W. Bialek (2000)

The information bottleneck method. arXiv preprint physics/0004057.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017)

R. Wang, B. Fu, G. Fu, and M. Wang (2017)

R. Wang, R. Shivanna, D. Z. Cheng, S. Jain, D. Lin, L. Hong, and E. H. Chi (2021)

L. Yu, W. Zhang, S. Zhou, T. Zhang, Z. Zhang, and D. Ou (2025)

Z. Zeng, X. Liu, M. Hang, X. Liu, Q. Zhou, C. Yang, Y. Liu, Y. Ruan, L. Chen, Y. Chen, Y. Hao, J. Xu, J. Nie, X. Liu, B. Zhang, W. Wen, S. Yuan, K. Wang, W. Chen, Y. Han, H. Li, C. Yang, B. Long, P. S. Yu, H. Tong, and J. Yang (2025)

Cited by: [§2.7.1](#).

Attention is all you need.

In Advances in neural information processing systems, pp. 5998–6008.

External Links: [Link](#)

Cited by: [§2.6.1](#).

Deep & cross network for ad click predictions.

In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, pp. 12:1–12:7.

Cited by: [§3.1](#).

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.

Cited by: [§3.1](#).

HHFT: hierarchical heterogeneous feature transformer for recommendation systems. arXiv preprint arXiv:2511.20235.

Cited by: [§1.1](#), [§3.2](#).

InterFormer: towards effective heterogeneous interaction learning for click-through rate prediction.

In Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), pp. 6225–6233.

Cited by: [§1.1](#).

J. Zhai, L. Liao, X. Liu, Y. Wang, R. Li, X. Cao, L. Gao, Z. Gong, F. Gu, J. He, Y. Lu, and Y. Shi (2024)

Actions speak louder than words: trillion-parameter sequential transducers for generative recommendations.

In Proceedings of the 41st International Conference on Machine Learning, pp. 58484–58509.

Cited by: [§1.1](#), [§1.2](#), [§3.2](#), [§4.1](#).

B. Zhang and R. Sennrich (2019)

Root mean square layer normalization.
arXiv preprint arXiv:1910.07467.

Cited by: [item 2](#).

B. Zhang, L. Luo, Y. Chen, J. Nie, X. Liu, S. Li, Y. Zhao, Y. Hao, Y. Yao, E. D. Wen, J. Park, M. Naumov, and W. Chen (2024)

Wukong: towards a scaling law for large-scale recommendation.

In Proceedings of the 41st International Conference on Machine Learning, pp. 59421–59434.

Cited by: [§3.1](#), [§4.1](#).

B. Zhang, L. Luo, X. Liu, J. Li, Z. Chen, W. Zhang, X. Wei, Y. Hao, M. Tsang, W. Wang, et al. (2022)

DHEN: a deep and hierarchical ensemble network for large-scale click-through rate prediction.

arXiv preprint arXiv:2203.11014.

Cited by: [§3.1](#).

Z. Zhang, H. Pei, J. Guo, T. Wang, Y. Feng, H. Sun, S. Liu, and A. Sun (2025)

OneTrans: unified feature interaction and sequence modeling with one transformer in industrial recommender.

arXiv preprint arXiv:2510.26104.

Cited by: [§1.1](#), [§3.2](#).

G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai (2019)

Deep interest evolution network for click-through rate prediction.

In Proceedings of the AAAI conference on artificial intelligence, Vol. **33**, pp. 5941–5948.

Cited by: [§3.2](#).

G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai (2018)

Deep interest network for click-through rate prediction.

In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining,

J. Zhu, Z. Fan, X. Zhu, Y. Jiang, H. Wang, X. Han, H. Ding, X. Wang, W. Zhao, Z. Gong, et al. (2025)

pp. 1059–1068.

Cited by: [§3.2](#).

Rankmixer: scaling up ranking models in industrial recommenders.

In Proceedings of the 34th ACM International Conference on Information and Knowledge Management,
pp. 6309–6316.

Cited by: [§1.1](#), [§3.2](#).

Experimental support, please [view the build logs](#) for errors. Generated by [L^AT_EX_{ML}](#) .

Instructions for reporting errors

We are continuing to improve HTML versions of papers, and your feedback helps enhance accessibility and mobile support. To report errors in the HTML that will help us improve conversion and rendering, choose any of the methods listed below:

- Click the "Report Issue" () button, located in the page header.

Tip: You can select the relevant text first, to include it in your report.

Our team has already identified [the following issues](#). We appreciate your time reviewing and reporting rendering errors we may not have found yet. Your efforts will help us improve the HTML versions for all readers, because disability should not be a barrier to accessing research. Thank you for your continued support in championing open access for all.

Have a free development cycle? Help support accessibility at arXiv! Our collaborators at LaTeXML maintain a [list of packages that need conversion](#), and welcome [developer contributions](#).