LIME: Link-based User-item Interaction Modeling with Decoupled XOR Attention for Efficient Test Time Scaling

Yang Liu[*†] **Yunjiang Jiang**[*] **Ayush Agarwal**[*] **Yihan Wu** **Haoran Liu** **Bi Xue**

Meta AI

{yangliu991, yunjiang, ayushaga, yihanwu, ryan0814, bixue}@meta.com

Abstract

Scaling large recommendation systems requires advancing three major frontiers: processing longer user histories, expanding candidate sets, and increasing model capacity. While promising, transformers' computational cost scales quadratically with the user sequence length and linearly with the number of candidates. This trade-off makes it prohibitively expensive to expand candidate sets or increase sequence length at inference, despite the significant performance improvements. We introduce **LIME**, a novel architecture that resolves this trade-off. Through two key innovations, LIME fundamentally reduces computational complexity. First, low-rank "link embeddings" enable pre-computation of attention weights by decoupling user and candidate interactions, making the inference cost nearly independent of candidate set size. Second, a linear attention mechanism, **LIME-XOR**, reduces the complexity with respect to user sequence length from quadratic ($O(N^2)$) to linear ($O(N)$). Experiments on public and industrial datasets show LIME achieves near-parity with state-of-the-art transformers but with a 10$\times$ inference speedup on large candidate sets or long sequence lengths. When tested on a major recommendation platform, LIME improved user engagement while maintaining minimal inference costs with respect to candidate set size and user history length, establishing a new paradigm for efficient and expressive recommendation systems.

1 Introduction

Modern recommendation systems operate at a massive scale, facing the challenge of ranking millions of candidate items within strict real-time latency constraints. To succeed, ranking models must navigate a fundamental trade-off between computational efficiency and predictive accuracy. This has led to two dominant but conflicting architectural paradigms. On one end of the spectrum is the two-tower model (Covington et al., 2016), which achieves unparalleled inference speed by encoding users and items into separate, independent representations. This separation enables the use of efficient approximate nearest-neighbor search (Shrivastava & Li, 2014), making real-time recommendations feasible at scale.

On the other end are powerful Transformer models like BST (Chen et al., 2019), which deliver state-of-the-art accuracy by encoding users' long interaction history (UIH) via multi-layer self attention and explicitly modeling deep, contextual interactions between that and each candidate item. Although this approach provides rich expressiveness, it comes with a significant computational cost: the self-attention over long sequences and the cross-attention between the full user sequence and

[*]Equal contribution.

[†]Project lead.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

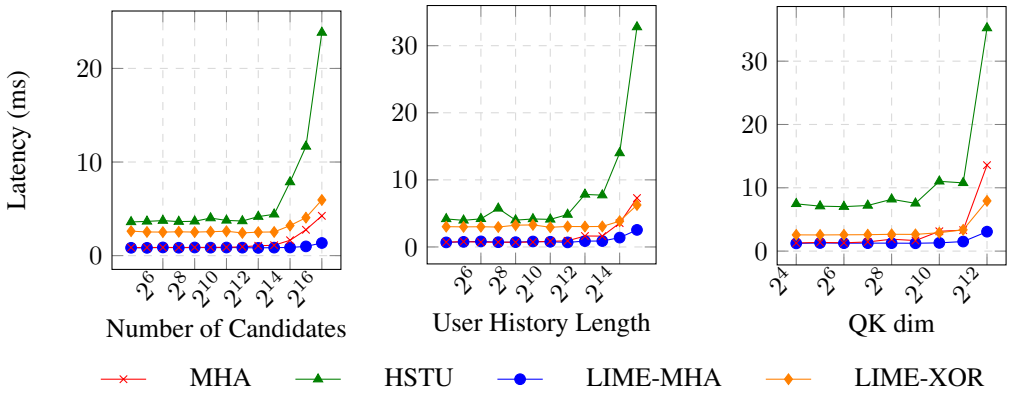


Figure 1: Overall latency analysis across different model parameters. This figure illustrates the inference latency vs number of candidates, user history length, and QK dimension of MHA, HSTU, LIME-MHA, and LIME-XOR. LIME-MHA consistently achieves the lowest and most stable latency across all three metrics.

each candidate become major performance bottlenecks. This architectural dilemma is becoming increasingly acute. The push for higher quality recommendations now demands scaling along three axes simultaneously: accommodating vast candidate sets, processing longer user histories, and deploying models of increasing complexity. These demands are fundamentally at odds, as existing efficient models lack expressiveness, while expressive models lack efficiency. Hybrid approaches (Li et al., 2022) offer only incremental improvements.

To fundamentally solve this problem and enable scaling along all three axes, we propose a new architectural blueprint: the Link-based User-Item Interaction Modeling for Efficient inference (LIME) framework. LIME is designed from the ground up to achieve the modeling power of a full cross-attention system while operating within the strict efficiency budget of a two-tower model. Its central innovation is a globally learned intermediate "link embedding" sequence that acts as a bridge between the long user history and candidate items. This design decouples the user and item representations during online inference, making scoring independent of history length by pre-computing the most expensive attention components offline. Furthermore, it enables the introduction of a new low rank attention mechanism to reduce user interaction history (UIH) self-attention complexity from quadratic to linear. By resolving these two primary computational bottlenecks, LIME provides a comprehensive solution to the expressiveness-versus-efficiency challenge, enabling deep interaction modeling at minimal latency (Figure 1).

Our primary contributions are as follows:

The Link Embedding Mechanism: We introduce a link embedding sequence that effectively approximates full cross-attention. This mechanism allows the expensive Query-Key attention weight, $\phi(\tilde{Q}\tilde{K}^\top)$, to be pre-computed and cached offline, enabling cross-attention-like expressiveness with the efficiency of Two-Tower modeling during online inference.

XOR Attention Masking: To overcome the quadratic time complexity of Transformers with respect to user history length, we propose an XOR attention mask that factorizes the full self-attention matrix into a bidirectional linear attention between the link embeddings and the user history sequence.

State-of-the-Art Performance and Impact: We demonstrate through extensive experiments that LIME achieves performance competitive with computationally intensive ranking models like HSTU (Zhai et al., 2024) with 10x lower latency. Deployed in production, LIME has yielded up to 38% source rate[3] gain on a major platform serving billions of users.

[3]This refers to the percentage of positively engaged items attributable to the ranking model using LIME.

2 Related Work

LIME addresses the long-standing trade-off between model expressiveness and inference efficiency in large-scale ranking. We situate our contributions in the context of two primary research areas: efficient ranking architectures and innovations in sequence modeling.

2.1 Efficient Ranking Architectures

The design of ranking models is dominated by a conflict between efficiency and interaction depth. On one end of the spectrum, two-tower models (Covington et al., 2016; Yi et al., 2019) achieve unparalleled efficiency. By encoding users and items into separate embedding spaces, they enable fast candidate retrieval using Approximate Nearest Neighbor (ANN) search. However, this separation prevents deep, feature-level interactions, limiting model expressiveness. On the other end, cross-attention models like DIN (Zhou et al., 2018) enable rich, target-aware interactions by dynamically attending to user history for each candidate, but their per-item computational cost makes them prohibitive for ranking large candidate sets.

Several approaches have sought to bridge this gap. Hybrid models add shallow interaction layers on top of a two-tower base (Li et al., 2022), though these often provide only marginal improvements. LIME offers a more fundamental solution. It introduces a novel bridge built upon a fixed set of global, learnable parameters, which we term link embeddings. These link embeddings act as an intermediary, allowing LIME to capture the rich dynamics of cross-attention while retaining the architectural efficiency of a two-tower model. This design, where a static, user-independent key space (raw links) facilitates the retrieval of information from a dynamic, personalized value space (personalized links), is conceptually similar to key-value memory networks (Miller, 2016) and the retrieval stage of retrieval-augmented models, enabling massive pre-computation. This approach directly tackles the sequential nature of user history and the need for scalable target-item interaction, striking a new balance between efficiency and expressiveness.

2.2 Efficient Attention for Long Sequence Modeling

The Transformer's quadratic complexity ($O(N^2)$) for self-attention remains a fundamental bottleneck for modeling long sequences. This challenge has driven the recent advances in Large Language Models (LLMs), moving beyond early approximations such as kernelization (Choromanski et al., 2020) or low-rank projections (Wang et al., 2020). More recent breakthroughs include new model classes—State Space Models (e.g., Mamba (Gu & Dao, 2024))—and hardware-aware techniques such as Lightning Attention (Qin et al., 2024) that directly optimize attention computation.

Within recommender systems, progress has often focused on adapting NLP-inspired attention mechanisms to click-through rate (CTR) prediction (Qi et al., 2022; Chen et al., 2022; Song et al., 2025) or using pruning to shorten sequences before attention is applied (Pi et al., 2020). While general-purpose mechanisms for handling set-based inputs exist, they often do not align with the specific needs of sequential recommendation. For instance, the Set Transformer (Lee et al., 2019) and its Pooling Multi-Head Attention (PMA) mechanism efficiently summarize a set into a fixed-size representation using learnable seed vectors. However, their primary goal is permutation-invariant summarization, whereas ranking requires target-aware representations that preserve the sequential nature of user histories.

LIME's XOR Attention contributes a distinct, task-specific solution. Rather than being a general-purpose approximation of the self-attention matrix, it is a mechanism co-designed with the LIME architecture. By using link embeddings as intermediaries, LIME structurally eliminates the need for direct history-to-history self-attention at inference time. This design enforces a linear complexity ($O(\ell \cdot N)$) tailored specifically for the user–item ranking context, representing a novel approach to building efficient and expressive sequence models for recommendations.

3 Model Overview

We propose the **L**ink-based user-item **I**nteraction **M**odeling for **E**fficient inference (**LIME**), a novel sequential-modeling architecture tailored for Click-Through Rate (CTR) prediction. LIME is designed to bridge the gap between highly efficient but less expressive two-tower models and powerful but

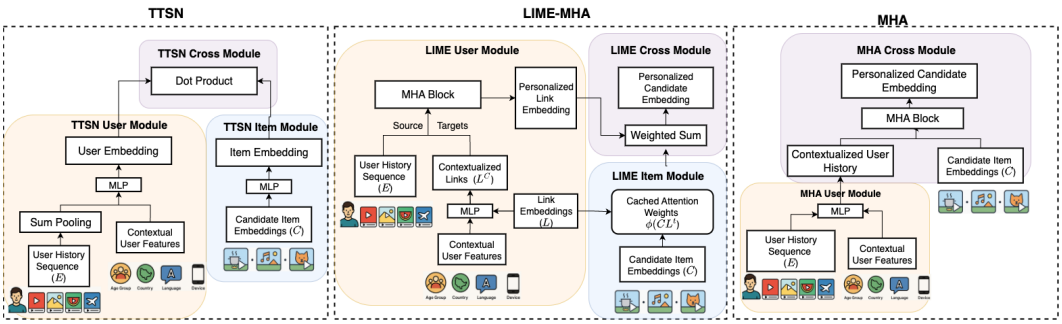


Figure 2: Architecture of TTSN (left), LIME-MHA (middle), and MHA (right). With a lightweight cross module using precomputed attention weights in a decoupled attention framework, LIME-MHA achieves MHA-level expressiveness with similar latency to TTSN.

computationally expensive cross-attention architectures. We present its design by progressively building from a simple, efficient baseline to a scalable model with deeper interactions.

To represent a user U's interaction history, we learn embedding tables to generate embeddings for each of the $N(U)$ items the user has interacted with. Each item is characterized by a set of attributes, which can be categorical (e.g., user action, topic id) or continuous (e.g., video length), and we learn embedding tables for each attribute. Continuous features are first transformed into categorical ones via bucketization to index into the embedding tables. For each item, we concatenate the learned embeddings of all its attributes and project them through a Multi-Layer Perceptron (MLP) to obtain a unified representation, E_j. The entire user history is then represented as a sequence of these embeddings, $E = \{E_j\}_{j=1}^{N(U)} \in \mathbb{R}^{N(U)\times d}$.

3.1 From Two-Towers to LIME-MHA: Bridging Efficiency and Expressiveness

While a two-tower model (TTSN) is highly efficient for scoring millions of items in retrieval, its expressivity is limited, as user-item interaction is confined to a late-stage dot product. At the other extreme, full cross-attention, prototyped by Multi-Head Attention (MHA), allows deep interaction between every candidate item and the entire user history. However, its computational cost, which scales with both candidate count ($M(U)$) and history length ($N(U)$), places the entire burden on the online interaction stage, making it prohibitively expensive.

LIME-MHA bridges this gap by introducing a small, fixed-size set of $\ell \ll N(U)$ auxiliary tokens, or **link embeddings** $L \in \mathbb{R}^{\ell\times d}$. These learned embeddings act as a compact summary of user interests, which are first personalized based on the user's history and then exposed to candidate items. This factorization is achieved via two MHA stages, effectively reducing the complexity from $O(M(U) \cdot N(U))$ to $O(\ell(M(U) + N(U)))$.

Multi-Head Attention (MHA), introduced by Vaswani et al. (Vaswani et al., 2017), is a function that maps q query vectors $Q \in \mathbb{R}^{q\times d}$ to outputs using n key and value vectors $K, V \in \mathbb{R}^{n\times d}$ as:

$$\mathrm{MHA}(Q, K, V; M; \theta) = \big(\phi(\tilde{Q}\tilde{K}^\top) \odot M\big)\tilde{V} \tag{1}$$

where $\tilde{Q} = QW_Q$, $\tilde{K} = KW_K$, $\tilde{V} = VW_V$, with learnable parameters $\theta = \{W_Q, W_K, W_V \in \mathbb{R}^{d\times d}\}$. Here, ϕ is an activation function (e.g., scaled Softmax or SiLU) and $M \in \{0, 1\}^{q\times n}$ is a binary mask. For intance, $J[i, j] :\equiv 1$ is the trivially all-1 mask pattern. $(A \odot B)_{ij} := A_{ij}B_{ij}$ stands for Hadamard product (elementwise multiplication).

The LIME-MHA architecture operates in two steps:

1. User-Side Link Personalization. The globally shared link embeddings L are first contextualized with user features E^C (e.g., location, device type) via an MLP:

$$L^C = \mathrm{MLP}(L \oplus E^C) \tag{2}$$

These contextualized links are then personalized by attending to the user's full interaction history E using a single MHA layer, producing personalized link embeddings L^P:

$$L^P = \mathrm{MHA}(L^C, E, E; J; \theta) \tag{3}$$

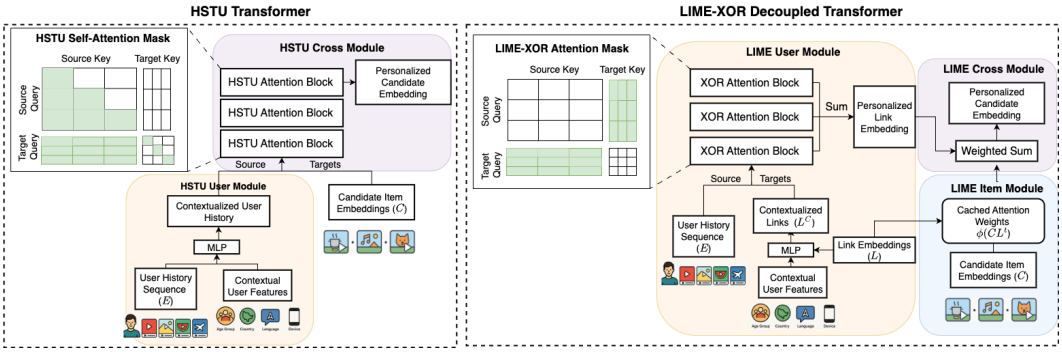


Figure 3: Architecture of HSTU Transformer (left) and LIME-XOR Decoupled Transformer (right) with a visual comparison of our proposed XOR attention mask against the standard HSTU causal self-attention mask.

2. Candidate-Side Decoupled Interaction. A key innovation of LIME is how candidate (target) embeddings T interact with the personalized links. Instead of a standard MHA where keys and values both come from L^P, we use the raw, user-independent link embeddings L as keys:

$$O = \text{MHA}(T, L, L^P; J; \theta) = \phi(TL^\top)L^P \tag{4}$$

This seemingly small change has a profound impact on efficiency. The attention weight matrix, $\phi(TL^\top) \in \mathbb{R}^{M(U)\times\ell}$, is now independent of the user. It can be pre-computed offline for all items in the corpus and cached. At inference time, this expensive matrix multiplication is replaced by a simple lookup, and the interaction reduces to a lightweight weighted sum of the personalized link embeddings L^P. This makes the serving latency per candidate effectively constant, $O(1)$, rather than scaling with history length.

3.2 From LIME-MHA to LIME-XOR: Scaling to Deeper Interactions

To enhance LIME's expressiveness, we can deepen the user-side module by stacking multiple interaction layers, akin to the architecture of a multi-layer Transformer. This computation is performed only once per user request, so its complexity does not impact the per-candidate scoring latency. A state-of-the-art approach for this would be to adapt a powerful sequential model block, such as the Hierarchical Sequential Transducer Unit (HSTU) (Zhai et al., 2024), which can be conceptually summarized as follows (see Appendix A for full notations):

$$\text{HSTUBlock}(X; \theta) = \text{GatedMLP}(\text{MHA}(X, X, X; M_{\text{causal}}; \theta); \eta) \tag{5}$$

where $X = E \oplus L^C$ is the concatenation of the user history and contextualized link embeddings. However, the standard HSTU attention uses causal self-attention mask, $M_{\text{causal}}[i, j] := i \geq j$, where every token in the user history attends to all preceding tokens (self-attention) and every candidate attends to the entire user history (cross-attention), results in a computational complexity of $O(N(U)^2 + N(U) \cdot M(U))$. This creates a major bottleneck for users with long interaction sequences or large candidate sets to rank.

To overcome this, we introduce XOR Attention (**XORA**), a novel attention kernel designed to replace the standard self-attention mechanism within the user-side module.

$$\text{XORA}(X, X, X; \theta) = \text{MHA}(X, X, X; M_{\text{xor}}; \theta) \tag{6}$$

$$= \text{MHA}(E, L^C, L^C; J; \theta) \oplus \text{MHA}(L^C, E, E; J; \theta) \tag{7}$$

where $M_{\text{xor}}[i, j] := 1_{i\in[0,|E|)} \wedge 1_{j\in[0,|E|)}$ is the exclusive-or mask pattern that ensures the source and target embeddings attend to one another only. As depicted in Figure 3, the XOR mask structurally eliminates the expensive history-to-history ($E \leftrightarrow E$) interactions. Instead, it facilitates an efficient, two-way, block-wise attention (7): the link embeddings attend to the user history ($L^C \rightarrow E$), and crucially, the user history (source) embeddings now also attend back to the link embeddings ($E \rightarrow L^C$).

This modification provides two main advantages. First, it reduces the computational complexity from quadratic to linear, $O(\ell \cdot N(U))$, making deep, multi-layer processing of long histories feasible. Second, it enriches the model's expressivity by enabling the user history representation to be modulated by the global context of the link embeddings from the very first layer.

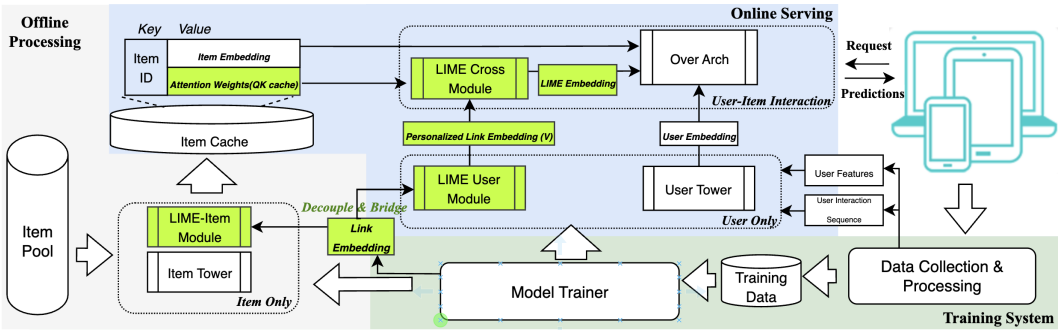


Figure 4: LIME Inference as part of the entire recommendation system

This leads to our advanced variant, **LIME-XOR**. In this model, we define an **XOR-Layer** by replacing the causal mask in (5) with our efficient XOR mask, M_{XOR} in (6). The personalized links are then computed by stacking and summing the outputs of n such layers:

$$L^P = \sum_{j=1}^{n} \text{GatedMLP}\left(\text{XORA}\left(E \oplus L^C, E \oplus L^C, E \oplus L^C; \theta\right); \eta_j\right) \tag{8}$$

This entire deep computation occurs on the user side, preserving the efficient, per-candidate scoring mechanism and LIME's overall scalability. Note that by using contextual links L^C as targets instead of candidate embeddings T, the process remains fully decoupled from individual candidates during this stage.

3.3 LIME's Architectural Advantages for Efficient Inference

The LIME framework is designed from the ground up to resolve the conflict between model expressiveness and inference latency. Its core efficiency stems from a strategic decoupling of user-side and item-side computations, bridged by the link embeddings. This design enables a multi-stage inference pipeline that maximizes offline pre-computation and minimizes online work, making it highly scalable for real-world recommendation systems.

To facilitate this efficiency, we employ a technique called **decoupled attention**, used in the final candidate interaction stage (Equation 4). Instead of using the user-specific personalized links (L^P) for both keys and values, we use the raw, user-independent link embeddings (L) as keys and the personalized links (L^P) as values. This asymmetric structure ensures that the expensive Query-Key dot product, $\phi(CL^t)$, depends only on item-side information (candidate embeddings C and raw links L). Consequently, this attention weight matrix can be pre-computed for all items in the corpus and stored in an efficient key-value store or index, such as FAISS (Johnson et al., 2017), effectively creating a cache of attention weights.

This architectural choice allows us to structure the entire inference process into three distinct stages, as illustrated in Figure 4.

Offline Item-Side Pre-computation. This stage is performed offline whenever the model or item is updated, eliminating redundant computation during online serving. Standard item features are processed by an item tower to produce item embeddings. The decoupled attention weights ($\phi(CL^\top)$) between all candidate item embeddings (C) and the raw link embeddings (L) are pre-computed and cached. This transforms the most intensive part of cross-attention into a simple lookup.

Online User-Side Computation. This stage runs once per user request and is independent of the number of candidates being scored. The user's context and interaction history are processed by the LIME user module (using either MHA or the multi-layer XOR Transformer) to produce the personalized link embeddings (L^P), as described in Equations (2)–(8). Crucially, this user-side computation can be executed in parallel with the candidate retrieval process. In a production environment, its latency is therefore largely masked, making even a deep, multi-layer Transformer on the user side feasible.

Lightweight User-Item Interaction. This final stage is executed online for each candidate but is extremely lightweight. For each candidate, the pre-computed attention weights are retrieved from the QK Cache. These weights are used to perform a simple weighted sum over the personalized link

embeddings (L^P) to generate the final LIME embedding. This embedding is then passed to a shallow interaction network for scoring.

By structuring inference this way, LIME achieves significant computational savings. Compared to a full cross-attention model like HSTU, which has a complexity of $\mathcal{O}(N(U)\cdot(N(U)+M(U)))$, LIME's complexity is reduced to $\mathcal{O}(\ell \cdot (N(U) + M(U)))$, where $\ell \ll N(U), M(U)$. More importantly, Model serving is reduced to a near-constant time operation with respect to the number of candidates, making LIME highly suitable for latency-sensitive ranking deployments with large candidate sets and long user histories.

4 Experimental Results

Baselines and Skylines. We compare our approach against the highly efficient **TTSN** (Two-tower sparse network; Figure 2) baseline, which sum-pools user history embeddings rather than using target attention. To establish performance upper-bounds (skylines), we evaluate two expressive but computationally prohibitive target-attention models: **MHA Skyline** (single-layer target attention) and **HSTU Skyline** (3 layers of causal self-attention and cross-attention between candidates and user history). For the details please refer Appendix D.1.

LIME Variants. We evaluate three LIME variants differing in their link personalization modules (full details in Appendix X). **LIME-MHA** (Eq. (3)) uses standard attention to pool user history into link representations. **LIME-XOR** (Eq. (8)) deepens contextualization using a 3-layer HSTU with our efficient XOR mask. Finally, **LIME-XOR+Window** extends the XOR mask with a linear sliding-window pattern (attending to ℓ local neighbors, akin to Longformer (Beltagy et al., 2020)) to jointly capture local and global interactions. To ensure a fair and controlled comparison, all other architectural components were held constant across all models (see details in Appendix D).

Metrics. We present the normalized entropy metrics (NE) (He et al., 2014) as well as (session) AUC metrics for all the variants and baselines/skylines. On the in-house industrial datasets, we measure accuracy on two tasks: video completion (VC) and watch time (WT). Normalized Entropy is defined for binary classification task as $\mathrm{NE}(\{(\mathrm{p_i}, \ell_i)\}_{\mathrm{i=1}}^{\mathrm{n}}) := \frac{\sum_{\mathrm{i=1}}^{\mathrm{n}} \ell_i \log \mathrm{p_i} + (1-\ell_i)\log(1-\mathrm{p_i})}{\log(\sum_{\mathrm{i=1}}^{\mathrm{n}} \ell_i / \mathrm{n})}$. Note that similar to logloss or binary cross entropy, lower NE means better accuracy. Usually 0.1% drop in NE will lead to online metric improvement.

4.1 Industrial & Public Experiments

Industry Experiments. We take 3 days of logged data for training, and 6 hours for evaluation. Compared to the public datasets, this is a larger scale dataset with longer user history sequences. As shown in Table 2, both LIME variants significantly outperform the TTSN baseline. LIME-MHA nearly matches the MHA skyline, while the multi-layer LIME-XOR closes the gap further, achieving performance competitive with the much more complex HSTU skyline across all tasks, even with a 32x sequence compression rate. The 1.04% VC NE improvement of LIME-XOR over the TTSN baseline is a significant gain, which translated to a +37.9% VC and +28.6% WT increase from LIME-ranked candidates during online A/B experiments (Table 2).

The online source-rate lifts in Table 2 are scoped to the source/generator traffic covered by the unified-preranker experiment and are not total platform-level lifts. In the same standard randomized experiment, the treatment group contained more than 10M users and the readout used a 7-day average. An engagement metric improved by $+0.111\%$ (90% CI $[0.041\%, 0.181\%]$), a watch-time metric by $+0.375\%$ (90% CI $[0.205\%, 0.545\%]$), and a session-level positive-engagement metric by $+0.233\%$ (90% CI $[0.080\%, 0.386\%]$). Broader watch-time and platform guardrails had no statistically significant regressions; capacity cost met the launch ROI bar.

Public Experiments. We benchmark on public datasets, namely Taobao Ads (Lyu et al., 2020) and KuaiRand-1K dataset (Gao et al., 2022). Taobao-Ads contains 25 million interactions with a maximum sequence length of 50 whereas KuaiRand-1K contains 12 million interactions with a maximum sequence length of 256. To unify data processing and evaluation framework, we leaned heavily on FuxiCTR, a comprehensive sequential recommendation model benchmark platform (Zhu et al., 2020). On both public datasets, LIME-MHA matches or outperforms the respective skyline with significant improvements over the sum-pooling baseline (Table 2 (b)). The improvement is

Table 1: Five-seed Taobao-Ad results (mean ± standard deviation). Lower log loss is better.

Model	gAUC	AUC	Log loss
VISTA (32 seeds)	0.569079 ± 0.000465	0.646075 ± 0.000224	0.192632 ± 0.000106
TTSN	0.570523 ± 0.000328	0.645368 ± 0.000216	0.192685 ± 0.000227
DIN	$\mathbf{0.571175 \pm 0.000489}$	0.647357 ± 0.000420	0.192587 ± 0.000458
HSTU	0.570691 ± 0.000500	$\mathbf{0.647669 \pm 0.000238}$	0.192597 ± 0.000170
LIME-XOR (32 links)	0.570567 ± 0.000152	0.646799 ± 0.000239	$\mathbf{0.192545 \pm 0.000111}$

Table 2: Offline, online, and public dataset experimental results.

(a) Industrial Results

Model	Offline VC NE	Offline WT AUC	Online A/B VC	Online A/B WT
TTSN	–	–	–	–
LIME-MHA	-0.72%	+0.46%	+28.6%	+22.1%
MHA Sky.	**-0.73%**	**+0.53%**	N/A	N/A
LIME-XOR	-1.04%	+0.76%	**+37.9%**	**+28.6%**
LIME-XOR+Window	**-1.11%**	**+0.80%**	N/A	N/A
HSTU Sky.	-1.06%	+0.77%	N/A	N/A

MHA and HSTU Skylines cannot be tested in online setting due to high serving latency from the large number of candidates.

(b) Public Dataset Results

Model	**KuaiRand-1K** Click AUC	**Taobao-Ad** AUC
TTSN	0.7389 (+0%)	0.6452 (+0%)
DIN	0.7404 (+0.20%)	0.6468 (+0.25%)
SASRec	0.7419 (+0.41%)	0.6462 (+0.15%)
Trunc. MHA	0.7351 (-0.51%)	0.6456 (+0.06%)
LREA	0.7408 (+0.26%)	0.6447(-0.08%)
LIME-MHA	**0.7433** (+0.60%)	**0.6465** (+0.20%)
MHA Sky.	0.7428 (+0.53%)	0.6464 (+0.19%)
LIME-XOR	0.7448 (+0.80%)	0.6467 (+0.23%)
LIME-XOR+Window	**0.7453** (+0.87%)	0.6468 (+0.25%)
HSTU Sky.	0.7444 (+0.74%)	**0.6475** (+0.36%)

largest in the KuaiRand-1K dataset as it contains longer sequences (length 256). On KuaiRand-1K, LIME-XOR performs comparably to the HSTU skyline but slightly worse on Taobao-Ad, though still outperforming LIME-MHA models. This is likely due to shorter sequence lengths in Taobao-Ad, leading to noisier results.

We also benchmark existing sequence-compression methods, including Truncated MHA, which applies the MHA skyline only to the most recent ℓ interacted items, and Linformer-style LREA (Song et al., 2025) with a low rank of ℓ. At the same low rank, LIME outperforms both compression techniques. Additionally we benchmark skyline models SASRec (Kang & McAuley, 2018) (a state-of-the-art transformer) and DIN (Deep Interest Network) (Zhou et al., 2018) to validate our LIME variants and skylines are indeed improvements over existing state-of-the-art models.

To quantify variance and compare with the concurrent VISTA architecture, we additionally ran five seeds on the same Taobao-Ad split and training protocol. Table 1 reports mean and standard deviation at a matched budget of 32 VISTA seed embeddings or LIME links. LIME-XOR improves all three mean metrics over our VISTA implementation, and its AUC improvement over VISTA and TTSN is statistically significant. Across all methods the comparison remains metric-dependent: DIN has the best gAUC, HSTU the best AUC, and LIME-XOR the lowest log loss.

4.2 Ablation Study & Scaling Laws

We ablate various components of LIME-XOR to demonstrate the effectiveness of its design by considering the following variants. **LIME-XOR w/ Link Pooling**: Instead of decoupled attention with candidate embeddings, we apply sum pooling to the personalized link embeddings. **LIME-XOR w/ Dot Product**: Instead of decoupled attention with candidate embeddings, we pass the ℓ dot products between candidate embedding and personalized link embeddings to the final prediction MLP. **LIME-MLP**: Instead of leveraging attention mechanisms, we directly learn MLPs to generate personalized low-rank embeddings (after padding the user history to fixed length) and cached weights. **LIME-XOR-MLP**: We preserve the LIME-XOR user module with multi-layer XOR attention and replace the LIME item module cached attention weights with weights learned from an MLP on the candidate item embeddings.

The ablation results in Table 4(a) demonstrate that both multi-layer XOR attention from the LIME user module and the decoupled attention mechanism in the LIME item module are critical to strong performance. The attention mechanisms cannot easily be replaced by MLP, sum pooling, or dot products in either user or item module especially due to variable user history sequence lengths. The

Table 3: Controlled five-seed Taobao-Ad link/key ablation with 32 links (mean $\pm$ standard deviation). Lower log loss is better.

Candidate-side basis	gAUC	AUC	Log loss	QK cacheable
Learned global links	0.570567 ± 0.000152	$\mathbf{0.646799 \pm 0.000239}$	$\mathbf{0.192545 \pm 0.000111}$	Yes
Frozen random links	0.570323 ± 0.000255	0.644434 ± 0.000268	0.192834 ± 0.000243	Yes
Fixed item centroids	$\mathbf{0.571714 \pm 0.000337}$	0.644832 ± 0.000288	0.192608 ± 0.000111	Yes
Personalized-link keys	0.569040 ± 0.000789	0.643578 ± 0.000553	0.192915 ± 0.000265	No

Table 4: Ablation Study and comparison to linear sparse attention variants on KuaiRand-1K dataset.

(a) Ablation Study

Model Variant	**AUC**
LIME-XOR	**0.7448**
LIME-XOR w/ Link Pooling	0.7400
LIME-XOR w/ Dot Product	0.7404
LIME-MLP	0.7407
LIME-XOR–MLP Hybrid	0.7442

(b) Comparison to SOTA Linear Attention Variants

Model Variant	**AUC**	**Complexity**
HSTU Skyline	0.7444	$O(N(U)^2 + N(U) \cdot M(U))$
Linear Transformer	0.7423	$O(N(U) \cdot M(U))$
Longformer	0.7443	$O(N(U) \cdot M(U))$
LREA (Linformer)	0.7408	$O(N(U) \cdot M(U))$
LIME-XOR	0.7448	$O(N(U)\ell + M(U)\ell)$
LIME-XOR+Window	**0.7453**	$O(N(U)\ell + M(U)\ell)$

controlled link/key ablation in Table 3 separately tests the candidate-side basis. Fixed item centroids have the best gAUC, while learned links retain the best AUC and log loss; the experiment therefore does not establish that learned anchors are uniformly superior to fixed alternatives. Personalized keys improve none of the three metrics and remove QK cacheability. For both NLP and CTR prediction tasks, transformer-based models (e.g., LLMs, HSTU) have shown improved performance as compute scales. On our large-scale ranking dataset, the decoupled transformer LIME-XOR similarly achieves substantial gains when scaling sequence length, link count, and model depth. Detailed results are provided in Appendix H.

4.3 Sparse Linear Attention Variants

We note that several alternate sparse linear attention mechanisms exist such as Linear Transformer (Katharopoulos et al., 2020), Longformer (Beltagy et al., 2020), and Linformer-style LREA (Wang et al., 2020; Song et al., 2025). These can be integrated into the skyline transformer to reduce the quadratic self-attention cost ($O(N(U)^2)$) to linear $O(N(U)\ell)$ for low-rank ℓ, though they will still suffer from $O(N(U) \cdot M(U))$ target attention cost. We can also integrate these linear attention variants within our proposed decoupled LIME framework to also optimize the target attention cost (e.g. LIME-XOR+Window).

From Figure 4(b), both Linear Transformer and Linformer-based methods have severe performance degradation but sliding-window based Longformer performs comparably to the HSTU skyline. However, it is still prohibitively expensive to serve (see Appendix K for latency analysis) due to the cross-attention cost. Yet, when we integrate sliding window attention into LIME-XOR we see improved performance with minimal latency, demonstrating the benefits of the LIME-XOR decoupled attention framework.

4.4 Inference Speed

LIME is highly scalable for ranking large candidate sets, an advantage for pre-ranking and retrieval. Figure 1 demonstrates that while MHA and HSTU latency grows significantly with more candidates or longer user histories, LIME's latency remains nearly constant (see Appendix G for a detailed comparison). This robustness makes it suitable for settings with very long sequences (e.g.,>30k items). Furthermore, the user backbone computation can be parallelized with candidate retrieval, masking most of its latency in production and yielding even greater savings than shown in these benchmarks.

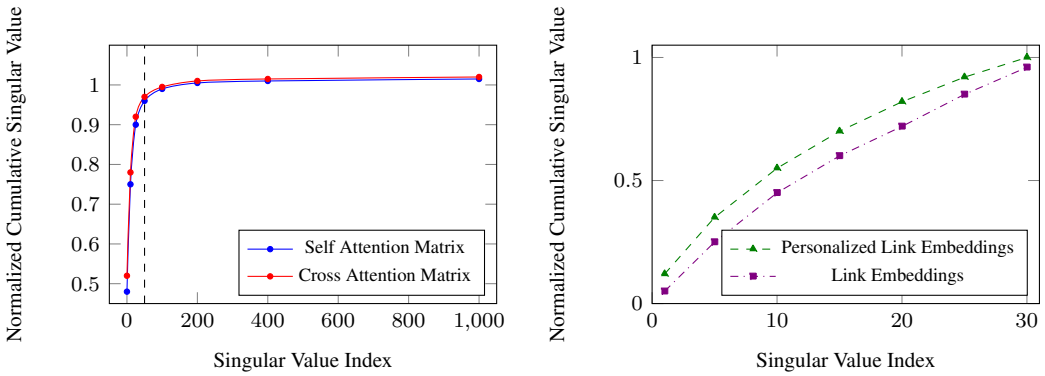


Figure 5: Left: cumulative singular value of self/cross attention matrices in a pretrained transformer model. Right: cumulative singular value of the raw/personalized link embeddings.

Table 5: Similarity between factorized LIME-MHA and direct MHA attention distributions.

Links	Jensen–Shannon divergence	Cosine similarity
1	0.122	0.837
4	0.072	0.902
32	0.055	0.915
128	**0.050**	**0.923**

5 Analysis and Discussions

LIME effectively projects the high-dimensional user-user and user-candidate interaction spaces into low-rank subspaces, acting as a surrogate attention mechanism. This decomposition can be viewed as a structured approximation of the full attention matrix, analogous to techniques in sparse/dense low-rank compression (Figure 6).

To this end, we compute the singular value decomposition (SVD) of the self-attention and cross-attention matrices in a trained HSTU Transformer skyline model averaged across layers and heads over 256 sequence-candidate pairs where each sequence has length 1024 and we rank 1024 candidates against the sequence. In Figure 5, the results clearly demonstrate a long-tailed pattern where the largest 32 singular values (denoted by the vertical black line) capture more than 90% of the information in both self and cross-attention matrices. We also compute the SVD of the raw link embeddings L and personalized link embeddings L^P of a trained LIME-XOR model and analyze the spectral distribution. The singular values of L and L^P demonstrate strong separation amongst the link embeddings with nearly full rank. This analysis indicates that 32 links are able to capture the majority of the information in the self-attention and cross-attention matrices with minimal redundancy for sequence lengths of 1024.

We also compare the factorized LIME-MHA attention distribution with direct MHA attention from independently trained models over 100k candidates and sampled users with history length 2,048. As the number of links increases from 1 to 128, Jensen–Shannon divergence decreases from 0.122 to 0.050 and cosine similarity increases from 0.837 to 0.923 (Table 5). For context, uniform attention has Jensen–Shannon divergence 0.693. The 32-link model already reaches 0.055 divergence and 0.915 cosine similarity.

6 Conclusion

We proposed LIME, a framework that resolves the efficiency-expressiveness trade-off in large-scale recommenders by using link embeddings as a low-rank approximation for target attention or more general transformer style sequence encoders. Experiments on public and industrial datasets show LIME matches the accuracy of state-of-the-art models like HSTU while drastically reducing inference latency, a result confirmed by significant online A/B test improvements. Our analysis confirmed that LIME's learned link embeddings effectively capture user interests, validating the low-rank hypothesis. By decoupling the user representation from target items, LIME offers a practical and powerful solution for building high-performance, scalable recommender systems.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Jianxin Chang, Chenbin Zhang, Zhiyi Fu, Xiaoxue Zang, Lin Guan, Jing Lu, Yiqun Hui, Dewei Leng, Yanan Niu, Yang Song, et al. Twin: Two-stage interest network for lifelong user behavior modeling in ctr prediction at kuaishou. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3785–3794, 2023.

Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 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, 2019.

Qiwei Chen, Yue Xu, Changhua Pei, Shanshan Lv, Tao Zhuang, and Junfeng Ge. Efficient long sequential user data modeling for click-through rate prediction. *arXiv preprint arXiv:2209.12212*, 2022.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

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.

Juechu Dong, Boyuan Feng, Driss Guessous, Yanbo Liang, and Horace He. Flex attention: A programming model for generating optimized attention kernels. *arXiv preprint arXiv:2412.05496*, 2024.

Chongming Gao, Shijun Li, Yuan Zhang, Jiawei Chen, Biao Li, Wenqiang Lei, Peng Jiang, and Xiangnan He. Kuairand: An unbiased sequential recommendation dataset with randomly exposed videos. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pp. 3953–3957, 2022.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the eighth international workshop on data mining for online advertising*, pp. 1–9, 2014.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus, 2017. URL `https://arxiv.org/abs/1702.08734`.

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

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Xiangyang Li, Bo Chen, HuiFeng Guo, Jingjie Li, Chenxu Zhu, Xiang Long, Sujian Li, Yichao Wang, Wei Guo, Longxia Mao, et al. Inttower: the next generation of two-tower model for pre-ranking system. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 3292–3301, 2022.

Ze Lyu, Yu Dong, Chengfu Huo, and Weijun Ren. Deep match to rank model for personalized click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 156–163, 2020.

A Miller. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 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, 2020.

Tao Qi, Fangzhao Wu, Chuhan Wu, and Yongfeng Huang. Fum: Fine-grained and fast user modeling for news recommendation, 2022. URL `https://arxiv.org/abs/2204.04727`.

Zhen Qin, Weigao Sun, Dong Li, Xuyang Shen, Weixuan Sun, and Yiran Zhong. Lightning attention-2: A free lunch for handling unlimited sequence lengths in large language models. *arXiv preprint arXiv:2401.04658*, 2024.

Anshumali Shrivastava and Ping Li. Improved asymmetric locality sensitive hashing (alsh) for maximum inner product search (mips). *arXiv preprint arXiv:1410.5410*, 2014.

Xin Song, Xiaochen Li, Jinxin Hu, Hong Wen, Zulong Chen, Yu Zhang, Xiaoyi Zeng, and Jing Zhang. Lrea: Low-rank efficient attention on modeling long-term user behaviors for ctr prediction. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2843–2847, 2025.

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.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 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.

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.

Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. Fuxictr: An open benchmark for click-through rate prediction. *CoRR*, 2020.

A Limitations and Future Work

Despite LIME's strong empirical performance, several limitations warrant discussion. First, LIME's effectiveness relies on the low-rank approximation assumption; domains with highly fragmented user interests may require higher-rank representations. Second, pre-computing QK caches requires periodic updates for evolving item catalogs, which makes it better suited for early stage ranking.

We are actively exploring the following directions: (1) building a larger pool of link embeddings and dynamically selecting link subsets based on user-side features or context, enabling personalized link allocation across different user segments, (2) extensions to multi-modal recommendation and cross-domain transfer learning to address the cold start user and item problem.

B Summary of Notations

We summarize the key notations used in the paper in Table 6.

Table 6: Notation Table

Symbol	Description
U	User
$N(U)$	Number of items in user U's interaction history
$M(U)$	Number of candidate items for user U
E_j	Embedding representation of item j
$E = \{E_j\}_{j=1}^{N(U)}$	Sequence of user history embeddings, $E \in \mathbb{R}^{N(U)\times d}$
E^C	User context features (e.g., location, device type)
T	Candidate (target) item embeddings
d	Embedding dimension
$L \in \mathbb{R}^{\ell\times d}$	Link embeddings (auxiliary tokens), $\ell \ll N(U)$
L^C	Contextualized link embeddings (personalized with user features)
L^P	Personalized link embeddings (after attention to user history)
$\mathrm{MHA}(Q, K, V; M; \theta)$	Multi-Head Attention function
W_Q, W_K, W_V	Learnable projection matrices for queries, keys, values ($\in \mathbb{R}^{d\times d}$)
θ, η	Set of learnable parameters
ϕ	Activation function (e.g., scaled Softmax, SiLU)
O	Final LIME output embedding after candidate-side MHA
M	Binary mask for attention, $M \in \{0,1\}^{q\times n}$
J	All-ones mask pattern, $J[i,j] \equiv 1$
M_{causal}	Causal self-attention mask, $M_{\text{causal}}[i,j] := i \geq j$
M_{xor}	XOR attention mask, $M_{\text{xor}}[i,j] := 1_{i\in[0,\lvert E\rvert)} \wedge 1_{j\in[0,\lvert E\rvert)}$
$\odot$	Hadamard (elementwise) product
$\oplus$	Concatenation operator
GatedMLP	MLP block with parameters η to learn a gated element-wise product
$\mathrm{HSTUBlock}(X;\theta)$	Hierarchical Sequential Transducer Unit block with parameters $theta$
XORA	XOR Attention Block
n	Number of stacked XOR layers

C Ranking Model Architecture Comparisons

Outlined in Table 7, two-tower based models have the highest scalability in both candidate set size and user history length but suffer from extremely limited (e.g. dot product) late interaction. Conversely, transformer-based models have deep user-item interactions through full self-attention (amongst user history) and cross-attention (between candidates and user history) at every layer. However, transformer-based models suffer from high latency and low scalability on both user history length and candidate set size.

SIM (Pi et al., 2020) and TWIN (Chang et al., 2023) improve scalability over Transformer-based methods through a two-stage approach. The general-search unit (GSU) first searches for the top-K relevant user history interactions for a particular candidate and exact-search unit (ESU) performs a full cross-attention against the retrieved items. While the latency is reduced compared to Transformer, we also sacrifice some model expressivity when we completely remove user history self-attention and the GSU runtime complexity is $O(MN)$ which can still be inefficient.

On the other hand, LIME is able to model deep user-item interactions (through the decoupled Transformer framework introduced in Section 3) and achieves high scalability to both candidate set size and user history length when $L \ll N$.

Table 7: Comparison of Ranking Model Architectures where N is the user history length, M is the number of candidates to rank, and $L \ll N$ is the LIME-compressed history size.

Axis	**Transformers (HSTU)**	**SIM/TWIN**	**Two-Tower**	**LIME (Ours)**
User-Item Interaction	Deep	Medium	Limited late interaction	Deep
Latency	High	Medium	Low	Low
Complexity	$O(MN + N^2)$	$O(MN)$	$O(N)$	$O(M\ell + N\ell)$
Pre-computation	Minimal	Minimal	Item Emb	QK Attention Weights

D Experiment Implementation Details

D.1 Baseline Setup

D.1.1 Baselines

We test multiple baseline models ranging from the simplest sum-pooling of user history embeddings to other sequence compression techniques listed below.

TTSN (Two-tower sparse network): Depicted in Figure 2, this baseline applies sum-pooling of all the user history embeddings instead of target attention with candidate embeddings.

D.1.2 Skylines

We cannot launch naive target attention (candidate against user history) directly for long user histories and large numbers of candidates to rank. But such powerful models can serve as our skyline goal on performance.

MHA Skyline computes a single layer of target attention between candidate items and user history items.

HSTU Skyline computes 3 layers of causal self attention and cross attention at each layer, between the candidate items and user history items.

D.1.3 LIME Variants

To thoroughly evaluate our proposed architecture, we conducted experiments with two primary variants of LIME that differ specifically in the sophistication of the link personalization module.

LIME-MHA: This is the fundamental implementation of our model, as described in Equation (3). It employs a standard attention mechanism to generate a personalized representation for each link by pooling information from the user's historical item embeddings.

LIME-XOR: This advanced variant, detailed in Equation (8), incorporates 3-layers of HSTU between the user history and link embeddings for deeper contextualization with our proposed efficient XOR-style masking.

LIME-XOR+Window: This variant extends the LIME-XOR attention mask by adding a sliding-window pattern over the user history, where each token attends to its ℓ neighbors on both sides (similar to Longformer (Beltagy et al., 2020)). The attention remains linear: each token attends to 2ℓ tokens—ℓ link tokens and ℓ local neighbors. XOR attention captures global interactions between user history tokens, while the windowed attention captures local interactions.

D.2 Dataset Preparation

For the Taobao Ad dataset, we leverage the preprocessed version provided by the FuxiCTR (Zhu et al., 2020) sequential modeling platform. For the KuaiRand dataset (Gao et al., 2022), we preprocess it ourselves by partitioning the first 14 days of interactions for training and last 2 days for testing. We also discarded all items from the train and test set with less than 30 total interactions for more reliable results. For the industrial dataset, we took 3 days of logged data for model training and 6 hours of data for evaluation.

D.2.1 Model Hyperparameters

For both public datasets and the industrial datasets, we fix model hyperparameters and seeds for all variants for a fair comparison (see Table 8). The industrial dataset is of the largest scale both in terms of the number of interactions and maximum sequence length. Due to the large size of the industrial dataset, we only train for a single epoch in a streaming single-pass setting for the industrial dataset.

Table 8: Model Hyperparameters for KuaiRand, TaoBao Ad, and Industrial datasets

Parameter	**KuaiRand**	**TaoBao Ad**	**Industrial**
Num. of Interactions	12×10^6	25×10^6	100×10^9
Learning Rate	1×10^{-4}	1×10^{-3}	4×10^{-4}
Batch Size	1024	8192	1024
Number of Heads	4	4	4
Epochs	2	10	1
Embedding Dimension	32	32	256
Number of Links	16	8	32
Max Sequence Length	256	50	1024
Interaction MLP	[512, 128, 64]	[512, 256, 128]	[96]

D.3 Model Design

Several additional design choices were crucial to stabilize training and ensure generalization:

- **Normalization:** We apply Layer normalization before each linear projection in the QKV projections. Without normalization, the embeddings can drift toward high magnitudes, which can collapse attention weights.
- **Link Initialization:** The raw link embeddings ℓ_i are initialized with samples from a standard normal distribution. This encourages diversity and allows the model to discover interpretable interest clusters during training.
- **Attention Function:** In single-layer MHA experiments we use scaled Softmax which performs the best and in multi-layer transformer experiments we leverage SiLU as the attention activation ϕ.
- **Training Objective:** We use a standard cross-entropy loss with multi-task objectives per request. Other loss terms (e.g., contrastive objectives or auxiliary disentanglement losses) may be explored in future work.

E Low Rank Expressions for Self-attention and Cross-attention

LIME effectively projects the high-dimensional user-user and user-candidate interaction spaces into low-rank subspaces, acting as a surrogate attention mechanism:

$$\mathrm{MHA}(T, E) \approx \mathrm{MHA}(T, L) \cdot \mathrm{MHA}(L, E),$$
$$\mathrm{MHA}(E, E) \approx \mathrm{MHA}(E, L) \cdot \mathrm{MHA}(L, E).$$

This decomposition can be viewed as a structured approximation of the full attention matrix, analogous to techniques in sparse/dense low-rank compression (e.g. Performer, Linformer, SVDNet).

F Information Bottleneck Perspective

From an information-theoretic viewpoint, LIME compresses the user history E into a set of link embeddings L, which are optimized to retain relevance to the user's behavior (via personalized attention pooling) and retain discriminative power for candidate (T) ranking.

This fits into the Information Bottleneck (IB) principle:

$$\min_{L} \mathcal{I}(L; E) - \beta \mathcal{I}(L; T)$$

where $\mathcal{I}$ denotes mutual information. That is, we retain only those aspects of the user history that are useful for predicting interaction with candidates.

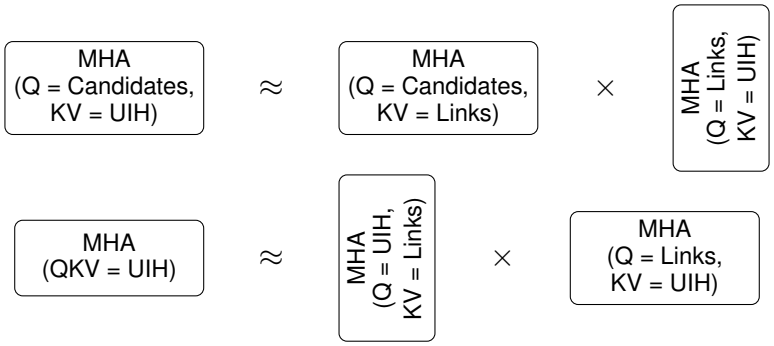


Figure 6: Top: Cross-attention as a low-rank product. Bottom: Self-attention as two low-rank cross-attentions. UIH stands for user interaction history.

G Inference Speed Comparison

Depicted in Figure 1 and Table 9, we observe that LIME-MHA and LIME-XOR have significantly lower latencies under large candidate sets to rank, long user histories, and high QK dimensions (due to the attention weight cacheing).

Candidate-side cache decomposition. We separately benchmark the 32-link candidate-side operator with batch size 1, history length 50, dimension 96, four heads, 20 warmups, and 100 CUDA-event repetitions on an NVIDIA PG509-210 Ampere GPU. Table 10 includes lookup and weighted aggregation rather than reporting lookup alone. BF16 caching yields a 1.04–$1.39\times$ P50 speedup over the uncached global-key operator in this eager module-level benchmark because aggregation remains the dominant cost. Eager int8 dequantization removes this gain at 100k candidates. At 10k candidates, BF16 lookup, weighted aggregation, and their combined execution have P50/P95 latencies of 0.038/0.054, 0.117/0.178, and 0.162/0.234 ms, respectively.

For a catalog $\mathcal{I}$, ℓ links, H heads, and b bytes per cached score, the current per-head layout occupies $|\mathcal{I}|\ell H b$ bytes. Thus 100M items, 32 links, four heads, and int8 scores require 12.8 GB; 3.2 GB would apply only to a head-aggregated layout. The current cache is rebuilt in model memory at each model publication (every five minutes), so lookup requires no network request. The observed miss rate is below 0.1%, and a staleness test changed online NE by less than 0.1% after several hours. Full production cache-build cost, miss-fallback latency, and deployed hardware/layout remain to be measured.

H Scaling Law for LIME-XOR Decoupled Transformer

Transformers (such as HSTU) have demonstrated strong scaling law for large-scale recommendation systems, mirroring scaling laws from LLMs from NLP. For our decoupled transformer, LIME-XOR, we test scaling across three axes on the large-scale industrial dataset: sequence length, number of links, and number of layers.

Table 9: Latency (in seconds) comparison for different models under varying conditions. The columns correspond to the models: LIME-MHA, MHA Skyline, HSTU Skyline, and LIME-XOR. All numerical values are truncated to two decimal places.

(a) vs. Number of Candidates (FlashAttention V2)

# Cand.	LIME-MHA	MHA Sky.	HSTU Sky.	LIME-XOR
16	0.84	0.87	3.60	2.62
32	0.83	0.87	3.66	2.53
64	0.88	0.88	3.74	2.52
128	0.84	0.87	3.64	2.55
256	0.87	0.88	3.67	2.52
512	0.85	0.88	4.01	2.56
1024	0.87	0.89	3.76	2.61
2048	0.85	0.95	3.70	2.43
4096	0.83	1.06	4.17	2.51
8192	0.86	1.11	4.42	2.53
16384	0.87	1.64	7.86	3.21
32768	0.99	2.76	11.66	4.06
65536	1.35	4.25	23.83	5.95

(b) vs. User History Length (FlashAttention V2)

Hist. Len.	LIME-MHA	MHA Sky.	HSTU Sky.	LIME-XOR
16	0.68	0.72	4.18	3.02
32	0.76	0.80	3.97	3.00
64	0.83	0.72	4.19	3.01
128	0.71	0.71	5.75	2.97
256	0.72	0.70	4.00	3.25
512	0.81	0.73	4.17	3.29
1024	0.77	0.85	4.11	2.94
2048	0.68	0.91	4.83	3.05
4096	0.87	1.63	7.83	3.04
8192	0.88	1.62	7.72	3.06
16384	1.39	3.56	14.00	3.84
32768	2.54	7.29	32.78	6.25

(c) vs. QK Dimension (PyTorch)

QK dim	LIME-MHA	MHA Sky.	HSTU Sky.	LIME-XOR
16	1.22	1.28	7.46	2.56
32	1.28	1.38	7.10	2.55
64	1.25	1.31	7.02	2.57
128	1.26	1.44	7.21	2.59
256	1.26	1.88	8.18	2.65
512	1.24	1.66	7.54	2.64
1024	1.30	3.13	11.02	2.90
2048	1.51	3.31	10.78	3.36
4096	3.07	13.57	35.23	7.93

Table 10: P50 candidate-side latency in milliseconds for 32 links. Cached totals include lookup and weighted aggregation.

Candidates	Uncached global	BF16 lookup	Weighted sum	BF16 total	Int8 lookup+dequant	Int8 total	Personalized keys
100	0.198	0.037	0.119	0.157	0.068	0.194	0.193
1,000	0.202	0.037	0.124	0.146	0.068	0.192	0.205
10,000	0.191	0.038	0.117	0.162	0.070	0.186	0.196
100,000	0.313	0.092	0.222	0.301	0.207	0.414	0.321

From Table 11, we observe that scaling sequence length is very effective in improving both VC NE and WT AUC. However, it is only effective under the presence of sufficient links. Scaling sequence length from 2k to 4k under 32 links is neutral but similar scaling with 256 links demonstrates significant NE and AUC wins. Moreover, for shorter sequences (e.g. 1k) scaling the number of links seems to reach an inflection point faster than scaling links for longer sequences.

From Table 13, we observe that scaling number of layers demonstrates significant NE and AUC improvements which helps demonstrate that LIME-XOR can also achieve scaling law up to the limits we were able to test.

	32	64	128	256
1k	0%	-0.04%	-0.19%	-0.20%
2k	-0.16%	-0.19%	-0.34%	-0.39%
4k	-0.11%	-0.36%	-0.36%	**-0.55%**

(a) VC NE % improvement across sequence lengths and link counts

	32	64	128	256
1k	0%	+0.01%	+0.08%	+0.11%
2k	+0.03%	+0.05%	+0.18%	+0.18%
4k	+0.03%	+0.13%	+0.17%	**+0.27%**

(b) WT AUC across sequence lengths and link counts

Table 11: Performance across varying sequence lengths (1k, 2k, 4k) and link counts (32, 64, 128, 256). Each cell reports metric value and relative change from the baseline (1k, 32 links).

Table 12 complements the quality scaling results with candidate-side latency and cache storage. At 10k candidates the eager kernels are largely launch-bound, so latency is not monotonic in ℓ, while storage grows linearly.

On the industrial dataset, we also compare LIME with geometric alternatives. Relative to LIME (NE change 0%, latency 1.39 ms), two-round and four-round LSH regress NE by $+0.50\%$ and $+0.43\%$ at 3.31 and 6.41 ms, respectively. Clustered attention with 128 and 1,024 centroids regresses NE by $+0.32\%$ and $+0.19\%$ at 1.72 and 2.85 ms. Positive NE changes are worse.

I Derivations for XOR Attention Kernel

I.1 Forward Pass

Let the output of the XOR-attention be $O = (O[S], O[T])$, where S, T stand for source and target respectively. In the context of LIME, source is the user history item embeddings, while target is the link embeddings. Similarly define $Q[S], Q[T], K[S], K[T], V[S], V[T]$ to be the source and target portion of the query, key, value embedding sequences.

$$O[T] = \phi(Q[T]K[S]^\top)V[S]$$
$$O[S] = \phi(Q[S]K[T]^\top)V[T]$$

Table 12: P50 latency at 10k candidates and estimated int8 cache size for 100M items and four heads.

Links	User encoding (ms)	Uncached global (ms)	BF16 cached total (ms)	Int8 cached total (ms)	Int8 cache (GB)
8	0.330	0.187	**0.143**	0.186	3.2
16	0.347	0.193	**0.154**	0.182	6.4
32	0.340	0.191	**0.162**	0.186	12.8
64	0.349	0.195	**0.151**	0.186	25.6
128	0.476	0.212	**0.157**	0.197	51.2

Number of Layers	VC NE % Improvement	WT AUC % Improvement
3	0%	0%
6	-0.25%	+0.12%
9	-0.34%	+0.18%
12	**-0.38%**	**+0.19%**

Table 13: VC NE and WT AUC for increasing model depth on fixed 1k sequence length with 32 links.

I.2 Backward Pass

Let $dO[T]$ denote an infinitesimally small change in $O[T]$, also known as the its differential. Similarly define $dO[S], dV[S], dV[T], dQ[S], dQ[T], dK[S], dK[T]$.

Trivially we have

$$\frac{\partial O[T]}{\partial Q[S]} = \frac{\partial O[T]}{\partial K[T]} = \frac{\partial O[T]}{\partial V[T]} = 0$$
$$\frac{\partial O[S]}{\partial Q[T]} = \frac{\partial O[S]}{\partial K[S]} = \frac{\partial O[S]}{\partial V[S]} = 0$$

The total differential of the loss function is given by

$$dL = \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[S]}\right)^\top dO[S]\right) + \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[T]}\right)^\top dO[T]\right).$$

Since $dO[S] = \phi(Q[S]K[T]^\top)dV[T]$ and $dO[T] = \phi(Q[T]K[S]^\top)dV[S]$,

$$dL_V = \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[S]}\right)^\top \phi(Q[S]K[T]^\top)dV[T]\right) + \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[T]}\right)^\top \phi(Q[T]K[S]^\top)dV[S]\right).$$

By comparing coefficient with the matrix chain rule $dL_{V[T]} = \mathrm{Tr}\left(\left(\frac{\partial L}{\partial V[T]}\right)^\top dV[T]\right)$, we see that

$$\frac{\partial L}{\partial V[T]} = \phi(K[T]Q[S]^\top)\frac{\partial L}{\partial O[S]}.$$

Similarly

$$\frac{\partial L}{\partial V[S]} = \phi(K[S]Q[T]^\top)\frac{\partial L}{\partial O[T]}.$$

Next for derivatives with respect to Q, we have

$$dL_{Q[S]} = \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[S]}\right)^\top \phi'(Q[S]K[T]^\top)V[T]K[T]^\top dQ[S]\right)$$
$$dL_{Q[T]} = \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[T]}\right)^\top \phi'(Q[T]K[S]^\top)V[S]K[S]^\top dQ[T]\right).$$

Hence

$$\frac{\partial L}{\partial Q[S]} = K[T]V[T]^\top \phi'(K[T]Q[S]^\top)\frac{\partial L}{\partial O[S]}$$
$$\frac{\partial L}{\partial Q[T]} = K[S]V[S]^\top \phi'(K[S]Q[T]^\top)\frac{\partial L}{\partial O[T]}.$$

Finally from

$$dL_{K[T]} = \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[S]}\right)^\top \phi'(Q[S]K[T]^\top)V[T]Q[S]^\top dK[T]\right)$$
$$dL_{K[S]} = \mathrm{Tr}\left(\left(\frac{\partial L}{\partial O[T]}\right)^\top \phi'(Q[T]K[S]^\top)V[S]Q[T]^\top dK[S]\right),$$

we get

$$\frac{\partial L}{\partial K[S]} = Q[T]V[S]^\top \phi'(K[S]Q[T]^\top)\frac{\partial L}{\partial O[T]}$$
$$\frac{\partial L}{\partial K[T]} = Q[S]V[T]^\top \phi'(K[T]Q[S]^\top)\frac{\partial L}{\partial O[S]}$$

J Triton PseudoCode

Algorithm 1 XOR Mask and Denominator Computation

Require: Query index i, key index j, number of sources n_s
Ensure: Binary mask m, normalization denominator d

```
is_src_q ← (i < n_s)
is_src_k ← (j < n_s)
m ← is_src_q ⊕ is_src_k          ▷ Attend iff exactly one is source
d ← is_src_q ? n_t : n_s         ▷ Normalize by opposite partition size
return m, d
```

Algorithm 2 Forward Pass with Block Range Selection

Require: $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$, number of sources n_s
Ensure: Output $\mathbf{O} \in \mathbb{R}^{n \times d}$

for $q_{\text{start}} \leftarrow 0$ to n step B_M **do** ▷ Parallel over query blocks
  $\mathbf{Q}_b \leftarrow \mathbf{Q}[q_{\text{start}} : q_{\text{start}} + B_M, :]$ ▷ Load to SRAM
  acc $\leftarrow \mathbf{0}_{B_M \times d}$

  // Block range selection (critical optimization)
  if $q_{\text{start}} + B_M \leq n_s$ **then** ▷ Pure source queries
    $k_{\text{lo}}, k_{\text{hi}} \leftarrow n_s, n$ ▷ Load target keys only
  else if $q_{\text{start}} \geq n_s$ **then** ▷ Pure target queries
    $k_{\text{lo}}, k_{\text{hi}} \leftarrow 0, n_s$ ▷ Load source keys only
  else ▷ Boundary case
    $k_{\text{lo}}, k_{\text{hi}} \leftarrow 0, n$
  end if

  for $k_{\text{start}} \leftarrow k_{\text{lo}}$ to k_{hi} step B_N **do** ▷ Over selected K,V blocks
    $\mathbf{K}_b \leftarrow \mathbf{K}[:, k_{\text{start}} : k_{\text{start}} + B_N]$ ▷ Load K,V to SRAM
    $\mathbf{V}_b \leftarrow \mathbf{V}[k_{\text{start}} : k_{\text{start}} + B_N, :]$
    $\mathbf{S} \leftarrow \mathbf{Q}_b \mathbf{K}_b$ ▷ Compute scores (tensor cores)

    for $i \leftarrow 0$ to B_M **do** ▷ Mask & normalize in registers
      for $j \leftarrow 0$ to B_N **do**
        $m, d \leftarrow \textsc{XorMask}(q_{\text{start}} + i, k_{\text{start}} + j, n_s)$
        $\mathbf{S}[i, j] \leftarrow m \ ?\ \text{SiLU}(\mathbf{S}[i, j])/d : 0$
      end for
    end for

    acc $\leftarrow$ acc $+ \mathbf{S}\mathbf{V}_b$ ▷ Accumulate (tensor cores)
  end for

  $\mathbf{O}[q_{\text{start}} : q_{\text{start}} + B_M, :] \leftarrow$ acc ▷ Write to HBM
end for
return O

Algorithm 3 Backward Pass (Transposed Access Pattern)

Require: $\mathbf{dO}, \mathbf{Q}, \mathbf{K}, \mathbf{V}$, number of sources n_s
Ensure: Gradients $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}$

$\mathbf{dQ}, \mathbf{dK}, \mathbf{dV} \leftarrow \mathbf{0}$
for $k_{\text{start}} \leftarrow 0$ to n step B_N **do** ▷ Parallel over K,V blocks
  $\mathbf{K}_b \leftarrow \mathbf{K}[:, k_{\text{start}} : k_{\text{start}} + B_N]$ ▷ Load K,V to SRAM (resident)
  $\mathbf{V}_b \leftarrow \mathbf{V}[k_{\text{start}} : k_{\text{start}} + B_N, :]$
  $\mathbf{dK}_{\text{acc}}, \mathbf{dV}_{\text{acc}} \leftarrow \mathbf{0}_{d \times B_N}, \mathbf{0}_{B_N \times d}$

  if $k_{\text{start}} + B_N \leq n_s$ **then** ▷ Source K,V block
    $q_{\text{lo}}, q_{\text{hi}} \leftarrow n_s, n$ ▷ Process target queries only
  else if $k_{\text{start}} \geq n_s$ **then** ▷ Target K,V block
    $q_{\text{lo}}, q_{\text{hi}} \leftarrow 0, n_s$ ▷ Process source queries only
  else ▷ Boundary case
    $q_{\text{lo}}, q_{\text{hi}} \leftarrow 0, n$
  end if

  for $q_{\text{start}} \leftarrow q_{\text{lo}}$ to q_{hi} step B_M **do** ▷ Over selected Q blocks
    $\mathbf{Q}_b \leftarrow \mathbf{Q}[q_{\text{start}} : q_{\text{start}} + B_M, :]$
    $\mathbf{dO}_b \leftarrow \mathbf{dO}[q_{\text{start}} : q_{\text{start}} + B_M, :]$
    $\mathbf{S} \leftarrow \mathbf{Q}_b \mathbf{K}_b$ ▷ Recompute forward (activation remat)

    for $i \leftarrow 0$ to B_M, $j \leftarrow 0$ to B_N **do**
      $m, d \leftarrow \textsc{XorMask}(q_{\text{start}} + i, k_{\text{start}} + j, n_s)$
      if m **then**
        $s_{ij} \leftarrow \text{SiLU}(\mathbf{S}[i, j])/d$
        $\mathbf{S}[i, j] \leftarrow s_{ij} \cdot (1 - s_{ij}) \cdot (1 + \mathbf{S}[i, j])/d$ ▷ SiLU gradient
      else
        $\mathbf{S}[i, j] \leftarrow 0$
      end if
    end for

    $\mathbf{dV}_{\text{acc}} \leftarrow \mathbf{dV}_{\text{acc}} + \mathbf{S}^\top \mathbf{dO}_b$ ▷ Accumulate (tensor cores)
    $\mathbf{dK}_{\text{acc}} \leftarrow \mathbf{dK}_{\text{acc}} + (\mathbf{S}^\top \mathbf{Q}_b)^\top$
    $\textsc{AtomicAdd}(\mathbf{dQ}[q_{\text{start}} : q_{\text{start}} + B_M, :], (\mathbf{dO}_b \mathbf{V}_b^\top)\mathbf{S}^\top)$
  end for

  $\mathbf{dK}[:, k_{\text{start}} : k_{\text{start}} + B_N] \leftarrow \mathbf{dK}_{\text{acc}}$ ▷ Write gradients
  $\mathbf{dV}[k_{\text{start}} : k_{\text{start}} + B_N, :] \leftarrow \mathbf{dV}_{\text{acc}}$
end for
return $\mathbf{dQ}, \mathbf{dK}, \mathbf{dV}$

K Linear Attention Performance & Latency Analysis

In Figure 7, we compare various linear attention methods (not decoupled so still more expensive than LIME) to both LIME variants and skyline HSTU. The results demonstrate that LIME variants consistently match (or exceed) HSTU skyline AUC under varying sequence length and QK-dimension settings.

We benchmark all variants using PyTorch's FlexAttention library (Dong et al., 2024) which enables us to flexibly compare various masking strategies, ensuring that unnecessary blocks of computation are skipped, without writing custom Triton logic for each type of linear attention. In Figure 8, we observe that Longformer is less expensive than Skyline HSTU when scaling user history length and number of candidates. However, it is still more expensive than LIME decoupled variants in complexity ($O(NM)$ vs $O(N\ell + M\ell)$). We observe it is more than twice as expensive across all user history lengths and spikes at user histories longer than 4096 and at number of candidates to rank greater than 2048.

Figure 7: AUC Comparison of Sparse Attention Variants with columns reporting on a particular Sequence Length/QK Dimension (e.g. 256/32 means 256 sequence length, 32 QK-dimension).

Variant	**256/128**	**64/128**	**256/32**	**64/32**	**Complexity**
HSTU Skyline	0.7444	0.7300	0.7425	**0.7296**	$O(N^2 + NM)$
Linear Trans.	0.7423	0.7292	0.7404	0.7278	$O(NM)$
Longformer	0.7443	0.7300	0.7425	0.7292	$O(NM)$
Linformer (LREA)	0.7408	0.7289	0.7406	0.7275	$O(NM)$
LIME-XOR	0.7448	0.7301	**0.7427**	0.7295	$O(N\ell + M\ell)$
LIME-XOR+Win.	**0.7453**	**0.7305**	0.7425	0.7292	$O(N\ell + M\ell)$

L Link Clustering Analysis

For one of the LIME-XOR models trained on the industrial dataset, we pick several links and examine the items that have the largest attention weights with that particular link. In doing so, we find that the clustered items for each link share strong semantic properties (e.g., video topic, length, etc.). Some examples include:

- **Link 2**: Long videos
- **Link 3**: Food & Cooking videos
- **Link 5**: Dog & Animal videos
- **Link 10**: Furniture & DIY videos
- **Link 30**: Sports videos
- **Link 31**: Foreign videos

M LIME Performance on Cold-Start Users

Although LIME is not explicitly designed for cold-start users, we observe that it delivers topline improvements for this group in online A/B testing that are comparable to those seen in the overall user cohort. To further understand this empirical behavior, we evaluate our pretrained HSTU skyline and LIME-XOR models on the cold-start subset of KuaiRand-1k. In this evaluation (see Table 14), we vary the maximum number of prior interactions a user may have to qualify as a cold-start user.

Table 14: LIME-XOR vs HSTU Performance (AUC) on the cold-start user cohort in KuaiRand-1k

Cold-Start User Threshold	LIME-XOR	HSTU
$N(U) \leq 4$	**0.7022**	0.7016
$N(U) \leq 8$	**0.7179**	0.7173
$N(U) \leq 16$	**0.7284**	0.7273
$N(U) \leq 32$	**0.7369**	0.7356

One hypothesis for why LIME leads to improved performance for cold-start users is that LIME compresses a user's history into a fixed length embedding (equal to number of links) regardless of whether the user is a power user or cold-start user. This normalization allows the model to extract useful patterns even from sparse histories, reducing variance and improving robustness compared to models like HSTU that depend more directly on the number of prior interactions.

N LLM Usage in Preparation

We used LLM significantly to polish the language of all sections in the paper, including the abstract and appendix sections. The tikz codes in Figure 1, 5, 6, and 8 were generated based on data points we collected ourselves. Most of the tables were also formatted by LLM.

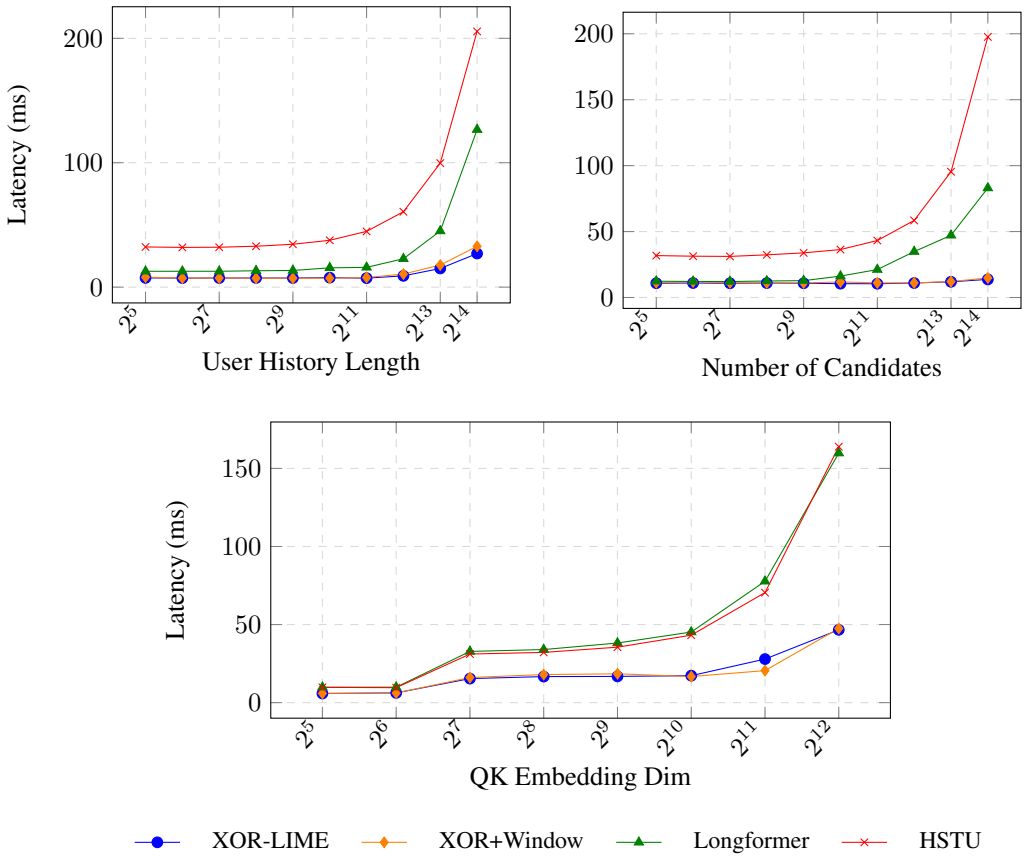


Figure 8: Latency comparison of linear attention to LIME-XOR and HSTU.

O Broader Impacts

Our work on the Link-based User-Item Interaction Modeling for Efficient inference framework presents several potential positive and negative societal impacts.

Positive Societal Impacts. LIME achieves state-of-the-art cross-attention expressiveness while operating within the strict efficiency budget of a two-tower model. The architecture delivers a 10x inference speedup over computationally intensive ranking models. When deployed on major platforms serving billions of users, this efficiency drastically reduces the computational resources required for large-scale recommendation systems. Besides, our empirical evaluations demonstrate that LIME delivers topline performance improvements for "cold-start" users who have limited prior interaction histories. By compressing a user's history into a fixed-length embedding regardless of their activity level, the model reliably extracts useful patterns from sparse data and reduces variance.

Potential Negative Impacts. Because LIME is highly effective at deep, contextual personalization based on long user histories , it carries the inherent risk of over-optimizing for user engagement. Highly accurate sequential recommendation models can inadvertently reinforce echo chambers and limit the overall diversity of content a user is exposed to.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: See Section A

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We provide the full set of assumptions and complete proof in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We fully disclose all the information needed to reproduce the main experimental results of the paper in the experimental section.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We will open source the code once the paper gets accepted.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the training and test details in our experimental section.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We don't report the error bars, but the online experimental results reported in our paper are statistical significance due to the large scale experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Due to the internal policy constraint, we are not allowed to list the computer resources needed to reproduce the experiments. We provide an efficiency analysis in Section 4.4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: See Section O.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper poses no such risks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The original owners of assets (e.g., code, data, models) used in the paper are properly credited.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: Due to the internal review constrain, we are not allowed to provide detailed information of the new assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: See Section N.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.