CLUEREC: LEARNING USER PREFERENCE EVIDENCE FOR RECOMMENDATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) have enhanced recommendation by bringing rich semantic knowledge into user and item modeling. However, existing LLM-based recommendation efforts mainly learn from positive interactions while overlooking the alternatives shown alongside each clicked item. Because a click is made within an exposure set, differences between the clicked item and its co-exposed alternatives provide valuable evidence about the user's relative preferences. Ignoring such evidence allows models to learn what users choose, but not why one item is preferred over another. To address this limitation, we propose **ClueRec**, an LLM-based recommendation model that explicitly learns preference evidence from exposure contexts. ClueRec comprises a Semantic Mapper, a Preference Evidence Miner and a Recommender. Its Semantic Mapper encodes multimodal item content into coarse-to-fine Semantic IDs (SIDs) and aligns the newly introduced SID tokens with the LLM's semantic space. The Preference Evidence Miner organizes clicked and co-exposed items into SID trees, extracts gated multi-level evidence from their divergent branches, and progressively injects this evidence into the LLM through attention. The Recommender utilizes this evidence-enhanced user representation to support both generative recommendation and CTR prediction. Extensive offline experiments on large-scale industrial data show that ClueRec consistently outperforms strong baselines on both generative recommendation and CTR prediction. Online A/B testing in a real-world advertising scenario further demonstrates stable improvements in user engagement and platform revenue. Our anonymous code is available at https://anonymous.4open.science/r/ClueRec-2E8B.

1 INTRODUCTION

LLM-based recommendation aims to bring the world knowledge and reasoning abilities of LLMs into personalized recommendation. Early studies (Wu et al., 2021; Hou et al., 2022) used pretrained language models to encoder item content and user behavior, thereby enhancing downstream recommenders. Later works (Geng et al., 2022; Cui et al., 2022) cast recommendation data and task objectives in a unified textual format, enabling retrieval and ranking through prompting or instruction tuning. Nevertheless, textual item descriptions can be lengthy or ambiguous. To address this issue, SID-based generative recommendation (GR) instead casts item prediction as autoregressive SID generation. TIGER (Rajput et al., 2023) demonstrated the feasibility of this paradigm by using an encoder–decoder architecture to generate SIDs. Recent works, such as OpenOneRec (Zhou et al., 2025) and PLUM (He et al., 2026), adopt pretrained LLMs as GR backbones and grounds newly introduced SID tokens through domain pretraining and SID–text alignment. The success of these efforts highlights the potential of combining SIDs with LLMs, marking a further step in the development of LLM-based recommendation.

*However, current LLM-based recommendation efforts ignored the **preference evidence** underlying user behavior*. In real systems, users interact with items from a set of exposed alternatives rather than in isolation. The differences between positively interacted items and their co-exposed alternatives—such as category, price, or brand—provide fine-grained evidence about user preferences. Yet current studies mainly learn from positive interactions without modeling these within-exposure comparisons. Consequently, they capture what users choose, but not why one item is preferred over its alternatives, limiting their understanding of user preferences.

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

To further illustrate the preference evidence, we present a SID-based example. SIDs are commonly constructed using residual quantization methods such as RQ-VAE, where earlier tokens encoding coarse semantics and later tokens capturing finer details. As shown in Figure 1, the SIDs of the clicked and co-exposed items can be organized into an SID tree based on their shared SID prefixes. The exposed items branch from the clicked item's path at different depths, reflecting semantic differences at coarse, intermediate, and fine levels. These branches provide preference evidence for the observed interaction, revealing why the clicked item was preferred over its co-exposed alternatives. Nevertheless, current LLM-based recommendation models learn only from the SID of the clicked item, omitting the branches formed by its co-exposed alternatives and thus failing to capture such preference evidence.

To address this issue, we propose **ClueRec**, a LLM-based recommendation model designed to learn preference evidence from user behaviors. Specifically, ClueRec consists of three components: a *Semantic Mapper*, a *Preference Evidence Miner*, and a *Recommender*. The *Semantic Mapper* constructs coarse-to-fine SIDs through residual quantization and aligns them with textual semantics, enabling the LLM to understand the hierarchical item semantics encoded in SIDs. The *Preference Evidence Miner* organizes clicked items and their co-exposed alternatives into SID trees based on shared prefixes. It identifies the first divergence between the clicked path and the exposure branches, from which it extracts multi-level preference evidence. The evidence is then aggregated by level-wise gating and injected into the LLM via attention to form an evidence-enhanced user representation. The *Recommender* leverages this user representation to support both generative recommendation and CTR prediction. We extensively evaluate ClueRec on a real-world platform serving hundreds of millions of users. Offline experiments and online A/B tests demonstrate that ClueRec consistently outperforms baselines on both tasks, validating its effectiveness in real-world recommendation.

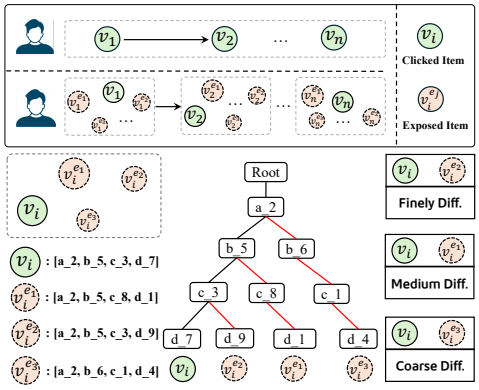

Figure 1: Illustration of the user preference evidence.

Overall, our main contributions are summarized as follows:

- **Preference Evidence Omission.** We identify the omission of preference evidence in existing LLM-based recommendation models. They mainly learn from isolated positive interactions while overlooking comparisons with co-exposed alternatives. As a result, they capture what users choose but struggle to understand why one item is preferred over others.
- **ClueRec Framework.** We propose ClueRec, which organizes clicked items and their co-exposed alternatives into SID trees. By contrasting the clicked path with exposure branches at different depths, it extracts multi-level preference evidence and injects it into the LLM, yielding an evidence-enhanced user representation for both generative recommendation and CTR prediction.
- **Empirical Results.** We conduct experiments on a large-scale industrial recommendation platform serving hundreds of millions of users. The results show that ClueRec delivers significant gains in offline and online recommendation performance. Comprehensive experiments further validate the effectiveness of preference evidence modeling and each key component of ClueRec.

2 PRELIMINARY

Concepts and Symbols. Let $\mathcal{U}$ and $\mathcal{I} = \{v_i\}_{i=1}^{|\mathcal{I}|}$ denote the user set and item set, respectively. The behavior history of user u is represented as $\mathcal{H}_u = [\mathcal{B}_1, \ldots, \mathcal{B}_N]$, where $\mathcal{B}_i = (v_i^+, \mathcal{E}_i)$ denotes the i-th impression event. Here, v_i^+ is the positively interacted item, and $\mathcal{E}_i = \{v_{i,1}^e, \ldots, v_{i,M}^e\}$ contains its M co-exposed alternatives. The positive interaction sequence is denoted by $\mathcal{S}_u^+ = [v_1^+, \ldots, v_N^+]$. In addition, We use $\mathcal{C}_u$ to denote the candidate item set in CTR, y_i to denote the interaction label of candidate v_i, and v_u^* to denote the target item.

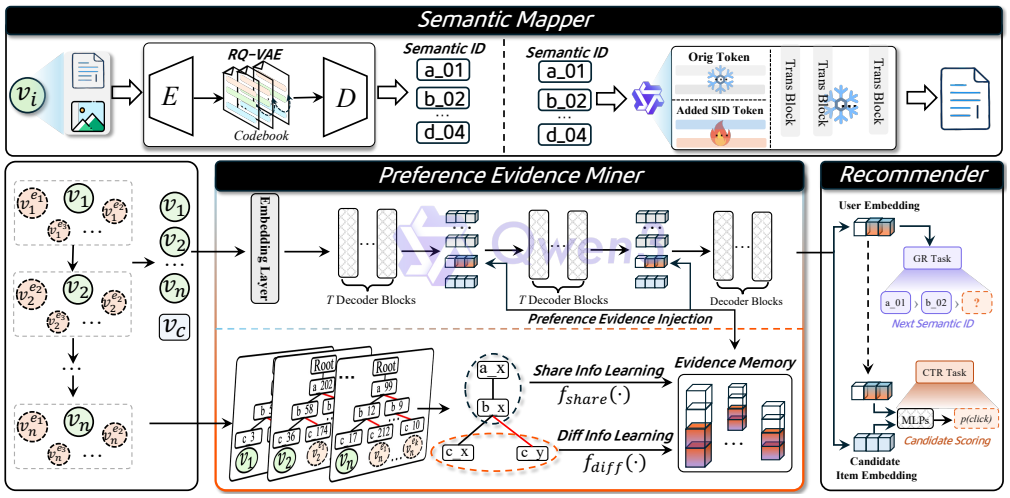

Figure 2: The framework of ClueRec, which consists of three components, includes Semantic Mapper, Preference Clue Miner, and Recommender.

Problem Formulation. Given the user history $\mathcal{H}_u$, our goal is to learn an evidence-enhanced user representation $\mathbf{u}_u$. We evaluate this representation learning framework in two task settings, including CTR prediction and generative recommendation. Specifically, CTR prediction estimates the interaction probability $\hat{y}_i$ for each candidate $v_i \in \mathcal{C}_u$ based on $\mathbf{u}_u$. Generative recommendation autoregressively generates the SID of the target item v_u^* conditioned on $\mathbf{u}_u$.

3 Methodology

As shown in Figure 2, ClueRec comprises three modules: a *Semantic Mapper*, an *Preference Evidence Miner*, and a *Recommender*. We introduce each of these modules in detail below.

3.1 Semantic Mapper

The Semantic Mapper aligns item SIDs with the semantic space of the LLM. It first encodes textual and visual information into coarse-to-fine hierarchical SIDs, and then uses knowledge injection to make the newly added SID tokens understandable to the LLM. This provides a unified item representation for subsequent preference modeling and recommendation. The module consists of two stages: SID Construction and SID Knowledge Injection.

SID Construction. For each item v_i, let x_i^t and x_i^v denote its textual description and image, respectively. We first use text encoder $f_t(\cdot)$ and image encoder $f_v(\cdot)$ of the pretrained Qwen3-VL-Embedding (Li et al., 2026) to extract semantic features and fuse (add operation) them into a multimodal item representation $\mathbf{x}_i \in \mathbb{R}^d$. We then apply RQ-VAE (Rajput et al., 2023) to perform L-level residual quantization:

$$z_i^l = \arg\min_{k \in \mathcal{K}^l} \left\| \mathbf{r}_i^l - \mathbf{e}_k^l \right\|_2^2, \quad \mathbf{r}_i^{l+1} = \mathbf{r}_i^l - \mathbf{e}_{z_i^l}^l, \quad \mathbf{h}_i = E_{rq}(\mathbf{x}_i), \quad \mathbf{x}_i = \mathrm{Fuse}\big(f_t(x_i^t), f_v(x_i^v)\big), \tag{1}$$

where $\mathbf{r}_i^1 = \mathbf{h}_i$, $\mathcal{K}^l$ denotes the l-th codebook, $\mathbf{e}_k^l$ is its k-th codeword, and $E_{rq}(\cdot)$ is the encoder of RQ-VAE. Following prior work, we train RQ-VAE using its standard objective. The indices selected from the five codebooks form the hierarchical SID of item v_i, i.e., $\mathbf{z}_i = [z_i^1, \cdots, z_i^L]$.

SID Knowledge Injection. To leverage the world knowledge and reasoning capabilities acquired by LLMs, we adopt Qwen 3.5 (Team, 2026) as the recommendation backbone. However, the constructed SIDs are absent from LLM's original vocabulary, preventing the model from directly understanding their item semantics. We therefore convert the SID codes at each level into new tokens and add them to LLM's vocabulary:

$$\mathcal{V}' = \mathcal{V} \cup \mathcal{V}_{\mathrm{SID}}, \qquad \mathcal{V}_{\mathrm{SID}} = \bigcup_{l=1}^{L} \{ z_i^l \mid v_i \in \mathcal{I} \}, \tag{2}$$

where $\mathcal{V}$ and $\mathcal{V}'$ denote the original and extended vocabularies of LLM, respectively. z_i^l is the SID token of item v_i at level l.

To enable the LLM to interpret the newly introduced SID tokens and understand their item semantics, we introduce an SID-to-text reconstruction task. Specifically, given an item SID $\mathbf{z}_i$, the model autoregressively reconstructs its textual description $\mathbf{w}_i = [w_{i,1}, \ldots, w_{i,N_i}]$. The objective is

$$\mathcal{L}_{\mathrm{KI}} = -\sum_i \sum_{j=1}^{N_i} \log p\left(w_{i,j} \mid \mathbf{z}_i, w_{i,<j}\right). \tag{3}$$

During training, we freeze all original LLM parameters and update only the newly added SID embeddings. This process aligns the SIDs with LLM's semantic space while preserving its pretrained knowledge and general capabilities. We adopt SID-to-text reconstruction as the knowledge injection objective. We discuss the choice of this objective in Appendix C.6.

3.2 Preference Evidence Miner

The Preference Evidence Miner aims to learn multi-level preference evidence by comparing clicked items with their co-exposed alternatives. For each impression, it organizes the clicked item and its co-exposed alternatives into an SID tree according to their shared prefixes. It then contrasts the clicked path with exposure branches at different depths to extract coarse-to-fine preference evidence. The extracted evidence forms an evidence memory, which is subsequently integrated into the LLM-derived user representation through attention. This module consists of two stages: Preference Evidence Learning and Preference Evidence Injection.

Preference Evidence Learning. Consider the i-th impression event in the user's history. The SIDs of the clicked item v_i^+ and the j-th exposed item $v_{i,j}^e$ are denoted as:

$$\mathbf{z}_i^+ = [z_i^{+,1}, \ldots, z_i^{+,L}], \qquad \mathbf{z}_{i,j}^e = [z_{i,j}^{e,1}, \ldots, z_{i,j}^{e,L}], \tag{4}$$

where earlier SID tokens capture coarse-grained semantics, while later tokens encode finer-grained item details. Thereafter, we identify the level at which the two SID paths diverge:

$$d_{i,j} = \min\left\{ l \mid z_i^{+,l} \neq z_{i,j}^{e,l} \right\}. \tag{5}$$

The two items therefore share the same SID prefix before level $d_{i,j}$. This shared prefix captures the semantic context in which the clicked and exposed items are compared. We obtain its representation $\mathbf{s}_{i,j} \in \mathbb{R}^d$ using a pooling function $f_{\mathrm{share}}(\cdot)$:

$$\mathbf{s}_{i,j} = f_{\mathrm{share}}\left(\mathbf{z}_i^{+,<d_{i,j}}\right) = \frac{1}{d_{i,j}-1} \sum_{r=1}^{d_{i,j}-1} \mathbf{e}_{z_i^{+,r}}, \tag{6}$$

where $\mathbf{e}_{z_i^{+,r}} \in \mathbb{R}^d$ denotes the LLM embedding of the SID token at level r. When $d_{i,j} = 1$, the two items have no shared prefix, and we set $\mathbf{s}_{i,j} = \mathbf{0}$.

Starting from level $d_{i,j}$, the two SID paths diverge into different semantic branches. For each $l \geq d_{i,j}$, we apply function $f_{\mathrm{diff}}(\cdot)$ to obtain the directed semantic difference at level l as:

$$\boldsymbol{\delta}_{i,j}^l = f_{\mathrm{diff}}\left(z_i^{+,l}, z_{i,j}^{e,l}\right) = \mathbf{e}_{z_i^{+,l}} - \mathbf{e}_{z_{i,j}^{e,l}}, \qquad l = d_{i,j}, \ldots, L. \tag{7}$$

Here, $\boldsymbol{\delta}_{i,j}^l \in \mathbb{R}^d$ points from the exposed branch to the clicked branch, representing the user's relative preference for the clicked item over the exposed alternative at semantic level l. Because subtraction is antisymmetric, reversing the order of comparison negates $\boldsymbol{\delta}_{i,j}^l$, allowing it to distinguish the observed preference $v_i^+ \succ v_{i,j}^e$ from its reverse at semantic level l.

A relative choice may be reflected at multiple semantic levels. To identify and aggregate the informative differences, we compute an independent gate for each level based on the shared semantic context $\mathbf{s}_{i,j} \in \mathbb{R}^d$, the current semantic difference $\boldsymbol{\delta}_{i,j}^l \in \mathbb{R}^d$, and the level information $\mathbf{e}_{\mathrm{level}}^l \in \mathbb{R}^d$:

$$\overline{\boldsymbol{\delta}}_{i,j} = \frac{1}{L - d_{i,j} + 1} \sum_{l=d_{i,j}}^{L} g_{i,j}^l \boldsymbol{\delta}_{i,j}^l, \qquad g_{i,j}^l = \sigma\big(\mathbf{w}_g^\top \tanh\big(W_g \left[\mathbf{s}_{i,j} \| \boldsymbol{\delta}_{i,j}^l \| \mathbf{e}_{\mathrm{level}}^l\right] + \mathbf{b}_g\big)\big), \tag{8}$$

where $W_g \in \mathbb{R}^{d\times 3d}$, $\mathbf{w}_g, \mathbf{b}_g \in \mathbb{R}^d$ are trainable parameters and $\mathbf{e}^l_{\text{level}} \in \mathbb{R}^d$ is the embedding of current level l. σ is the Sigmoid function. The output $\overline{\boldsymbol{\delta}}_{i,j} \in \mathbb{R}^d$ aggregates these level-wise differences and represents the multi-level preference direction from the exposed item $v^e_{i,j}$ to the clicked item v^+_i. Since preference evidence at different levels can coexist and complement one another, we use independent sigmoid gates instead of softmax competition across levels.

Preference Evidence Injection. We use LLM injected with SID knowledge as the backbone for user preference modeling. It learns the user's general preference from the click sequence, while the multi-level preference differences produced by the preceding stage capture fine-grained evidence from clicked–exposed comparisons. To fuse these two types of information, we first organize all preference evidences of user u into an evidence memory $\mathbf{M}_u$:

$$\mathbf{M}_u = \left[\mathbf{s}_{i,j} + \overline{\boldsymbol{\delta}}_{i,j}\right]_{(i,j)\in\mathcal{A}_u} \in \mathbb{R}^{P_u\times d}, \qquad \mathcal{A}_u = \{(i,j) \mid 1 \le i \le N,\ 1 \le j \le M_i\}, \tag{9}$$

where $M_i = |\mathcal{E}_i|$ denotes the number of co-exposed alternatives in the i-th impression, and $P_u = \sum_{i=1}^{N} M_i$ is the number of preference evidences for user u. Each $\overline{\boldsymbol{\delta}}_{i,j}$ represents the user's preference direction from the exposed alternative $v^e_{i,j}$ toward the clicked item v^+_i.

For the clicked-item sequence $\mathcal{S}^+_u = [v^+_1, \ldots, v^+_N]$, we serialize the corresponding SIDs into the token sequence $\mathbf{Z}^+_u = [\mathbf{z}^+_1; \ldots; \mathbf{z}^+_N]$. We then append a learnable $\langle\text{USER}\rangle$ token to obtain $\widetilde{\mathbf{Z}}^+_u = [\mathbf{Z}^+_u; \langle\text{USER}\rangle]$. Its hidden state represents the user's general preference. LLM learns the user's click behavior via a stack of decoder blocks, while preference evidence provides comparative context for interpreting these clicks. To help LLM progressively understand the user's clicks, we allow the evolving user representation to interact with the evidence memory at multiple intermediate layers.

Specifically, following the hybrid architecture of Qwen3.5, we divide its decoder layers into consecutive stages, each consisting of several linear-attention layers followed by a full-attention layer. We perform one evidence interaction immediately before the full-attention layer in each stage. Let $\mathbf{h}^{(T)}_{u,\text{user}}$ denote the hidden state of the $\langle\text{USER}\rangle$ token at the T-th interaction. We use it as the Query and the evidence memory as the Key and Value:

$$\mathbf{q}^{(T)}_u = \text{RMSNorm}^{(T)}\left(\mathbf{h}^{(T)}_{u,\text{user}}\right) W^{(T)}_Q, \tag{10}$$

$$\mathbf{K}^{(T)}_u = \text{RMSNorm}^{(T)}\left(\mathbf{M}_u\right) W^{(T)}_K, \quad \mathbf{V}^{(T)}_u = \text{RMSNorm}^{(T)}\left(\mathbf{M}_u\right) W^{(T)}_V. \tag{11}$$

The retrieved preference evidence is computed as:

$$\mathbf{c}^{(T)}_u = \text{softmax}\left(\frac{\mathbf{q}^{(T)}_u \left(\mathbf{K}^{(T)}_u\right)^\top}{\sqrt{d_h}}\right) \mathbf{V}^{(T)}_u W^{(T)}_O. \tag{12}$$

Here, $\mathbf{c}^{(t)}_u \in \mathbb{R}^d$ is the preference evidence retrieved at the T-th interaction, which aggregates the memory entries most relevant to the current user representation $\mathbf{h}^{(T)}_{u,\text{user}}$. The attention-head dimension is denoted by d_h, and the head index is omitted for simplicity. We reuse the normalization operation ($\text{RMSNorm}(\cdot)$) and the $Q/K/V/O$ projection matrices from the corresponding full-attention layer of Qwen 3.5. This allows the user representation and preference evidence to interact in LLM's native attention space. Since the evidence memory is unordered, no positional encoding or causal mask is applied during the interaction.

We then add the retrieved evidence to the $\langle\text{USER}\rangle$ representation through a residual connection, i.e., $\widetilde{\mathbf{h}}^{(t)}_{u,\text{user}} = \mathbf{h}^{(t)}_{u,\text{user}} + \mathbf{c}^{(t)}_u$. The updated $\langle\text{USER}\rangle$ token is processed by the next group of decoder blocks and interacts with the evidence memory again at the next group boundary. In this way, LLM progressively incorporates preference evidence to understand the user's clicks.

After the final decoder block, we use the resulting $\langle\text{USER}\rangle$ representation as the evidence-enhanced user representation $\mathbf{u}_u \in \mathbb{R}^d$.

3.3 Recommender

ClueRec provides a common preference-evidence modeling architecture for different recommendation objectives. To evaluate its cross-task applicability, we instantiate and train ClueRec separately for generative recommendation and CTR prediction.

Generative Recommendation. Given the user representation $\mathbf{u}_u$, the model generates the SID of the target item v_u^*, denoted by $\mathbf{z}_u^* = \left[z_u^{*,1}, \ldots, z_u^{*,L}\right]$. We then apply LLM to autoregressively generate SID tokens. The generation probability is factorized as:

$$p\left(\mathbf{z}_u^* \mid \mathbf{u}_u\right) = \prod_{l=1}^{L} p\left(z_u^{*,l} \mid \mathbf{u}_u, \mathbf{z}_u^{*,<l}\right), \tag{13}$$

where $\mathbf{z}_u^{*,<l}$ denotes the previously generated SID prefix. The model is trained with the standard autoregressive cross-entropy loss:

$$\mathcal{L}_{\mathrm{GR}} = -\sum_{l=1}^{L} \log p\left(z_u^{*,l} \mid \mathbf{u}_u, \mathbf{z}_u^{*,<l}\right). \tag{14}$$

At inference time, we construct a catalog-wide prefix tree from all valid item SIDs and perform constrained decoding over the tree.

Click-Through Rate Prediction. For a candidate item $v_c \in \mathcal{C}_u$ with SID $\mathbf{z}_c = [z_c^1, \ldots, z_c^L]$, we append its SID sequence and a learnable $\langle\mathrm{ITEM}\rangle$ token after the $\langle\mathrm{USER}\rangle$ token:

$$\widetilde{\mathbf{Z}}_{u,c}^{\mathrm{CTR}} = \left[\mathbf{Z}_u^+; \langle\mathrm{USER}\rangle; z_c^1, \ldots, z_c^L; \langle\mathrm{ITEM}\rangle\right]. \tag{15}$$

At each evidence-injection layer, only the $\langle\mathrm{USER}\rangle$ state attends to the evidence memory $\mathbf{M}_u$; the candidate tokens do not interact with it directly. Under causal attention, the candidate cannot affect the preceding user representation, whereas the $\langle\mathrm{ITEM}\rangle$ token can aggregate both the evidence-enhanced user context and the candidate semantics. Therefore, we take the final hidden state of the $\langle\mathrm{ITEM}\rangle$ token as the candidate representation $\mathbf{v}_i \in \mathbb{R}^d$.

We then feed the user representation, candidate representation, and their element-wise interaction into a task-specific CTR head:

$$\widehat{y}_i = \sigma\left(\mathrm{MLP}_{\mathrm{CTR}}\left([\mathbf{u}_u \| \mathbf{v}_i \| \mathbf{u}_u \odot \mathbf{v}_i]\right)\right), \tag{16}$$

where $\widehat{y}_i$ is the predicted probability that user u interacts with candidate v_i, $\|$ denotes concatenation, and $\odot$ denotes element-wise multiplication. We use the standard binary cross-entropy loss:

$$\mathcal{L}_{\mathrm{CTR}} = -\sum_{v_i \in \mathcal{C}_u} \left[y_i \log \widehat{y}_i + (1 - y_i) \log(1 - \widehat{y}_i)\right], \tag{17}$$

where $y_i \in \{0, 1\}$ is the interaction label of candidate v_i.

Mixed-Data Training. To preserve the LLM's general language capabilities and world knowledge while learning recommendation tasks and SID semantics, we train it on a mixture of recommendation data, general-domain QA data, and SID knowledge-injection data. Detailed data sources and mixing ratios are provided in Appendix C.1.

4 Experiments

4.1 Experimental Settings

Datasets. We conduct offline evaluation and online A/B testing on the same real-world large-scale recommendation platform. The offline dataset is randomly sampled from the platform logs of 5 million users over 7 consecutive days. Each record contains the user exposure, clicks, and co-exposed candidate items. We adopt a leave-one-out split: for each user, the last impression event is used for testing, the second-to-last for validation, and the remaining events for training. The online experiment is conducted as an A/B test on the same platform with traffic from tens of millions of users. Detailed statistics of the offline dataset are provided in Appendix C.1.

Baselines. We compare ClueRec with representative baselines for generative recommendation and CTR prediction. For generative recommendation, the baselines cover conventional sequential recommendation (SASRec (Kang & McAuley, 2018)), SID-based generation (TIGER (Rajput et al., 2023), LC-Rec (Zheng et al., 2024), and OpenOneRec (Zhou et al., 2025)), exposure-aware sequence modeling (CaseRec (Zhao et al., 2025)), and LLM reasoning-enhanced generation (CARE (Lin et al., 2026)). For CTR prediction, the baselines cover feature interaction

(DeepFM (Guo et al., 2017)), behavior sequence modeling (DIEN (Zhou et al., 2019) and SIM (Pi et al., 2020)), exposure-aware modeling (RACP (Fan et al., 2022)), and LLM-enhanced prediction (CoLLM (Zhang et al., 2025) and LLaCTR (Cui et al., 2026)). Detailed descriptions are provided in Appendix C.3.

Evaluation Metrics. For generative recommendation, we report Recall@N and NDCG@N, where $N \in \{20, 50\}$. The model generates the SID of the target item under constrained decoding. For CTR prediction, we report AUC, GAUC, and LogLoss.

Table 1: Overall performance comparison on generative recommendation and CTR prediction. All experiments are repeated five times, and the average results are reported. The best and second-best results are highlighted in bold and underlined, respectively. † denotes statistical significance under a paired t-test ($p < 0.05$).

Methods	Generative Recommendation				Mothods	CTR Prediction		
	Recall@20↑	NDCG@20↑	Recall@50↑	NDCG@50↑		AUC↑	GAUC↑	LogLoss↓
SASRec(2018)	0.3846	0.2491	0.4055	0.2534	DeepFM(2017)	0.7190	0.6775	0.4056
TIGER(2023)	0.4026	0.2523	0.4934	0.2701	DIEN(2019)	0.7385	0.7040	0.3912
LC-Rec(2024)	0.4088	0.2549	0.5146	0.2776	SIM(2020)	0.7412	0.7081	0.3894
CaseRec (2025)	0.4092	0.2533	0.5108	0.2752	RACP(2022)	0.7427	0.7092	0.3886
OpenOneRec(2025)	0.4123	0.2547	0.5224	0.2788	CoLLM(2025)	0.7492	0.7134	0.3845
CARE(2026)	0.4126	0.2530	0.5228	0.2775	LLaCTR(2026)	0.7478	0.7116	0.3858
ClueRec	**0.4218**†	**0.2612**†	**0.5379**†	**0.2847**†	**ClueRec**	**0.7545**†	**0.7213**†	**0.3807**†

4.2 Overall Performance

Table 1 reports the overall performance of ClueRec and all baselines on generative recommendation and CTR prediction. ClueRec achieves the best results on all metrics, and its improvements over the strongest baseline are statistically significant. For generative recommendation, ClueRec outperforms all baselines on Recall@N and NDCG@N ($N = 20, 50$). For CTR prediction, it also achieves the best AUC, GAUC, and LogLoss. These results show that ClueRec improves both generative recommendation and CTR prediction.

The advantage of ClueRec mainly comes from explicit modeling of preference evidence. In generative recommendation, the model extracts multi-level preference clues by comparing clicked items with co-exposed candidates, leading to more accurate SID generation. In CTR prediction, these clues are injected into the user representation via attention, enabling finer-grained discrimination among candidates. Both tasks are trained separately but share the same LLM backbone and SID knowledge, and both benefit from the preference evidence extracted from exposure contexts. These results indicate that modeling preference evidence from exposure context is essential for improving the overall capability of recommendation models.

Table 2: Ablation study of ClueRec. All experiments are repeated five times, and the average results are reported. † denotes statistical significance under a paired t-test ($p < 0.05$).

Methods	Generative Recommendation				CTR Prediction		
	Recall@20↑	NDCG@20↑	Recall@50↑	NDCG@50↑	AUC↑	GAUC↑	LogLoss↓
w/o PEM	0.4023	0.2478	0.5152	0.2701	0.7481	0.7134	0.3904
w/o KI	0.4045	0.2477	0.5168	0.2712	0.7452	0.7105	0.3932
w/o SPE	0.4186	0.2602	0.5354	0.2822	0.7527	0.7194	0.3820
w/o MLPE	0.4172	0.2581	0.5324	0.2806	0.7520	0.7172	0.3836
w/o MSI	0.4185	0.2601	0.5351	0.2829	0.7531	0.7198	0.3817
ClueRec	**0.4218**†	**0.2612**†	**0.5379**†	**0.2847**†	**0.7545**†	**0.7213**†	**0.3807**†

4.3 Ablation Study

To evaluate the contribution of each component in ClueRec, we consider the following variants:

- **w/o Preference Evidence Miner (PEM)**: It removes the entire preference evidence module and directly feeds both clicked items and exposed candidates into the original Qwen backbone.
- **w/o SID Knowledge Injection (KI)**: This model removes the SID-to-text reconstruction task and randomly initializes the newly added SID tokens without semantic alignment.

- **w/o Structured Preference Evidence (SPE)**: It replaces hierarchical difference modeling with an MLP over concatenated clicked–exposed SID embeddings, retaining all SID levels and the same evidence injection mechanism.
- **w/o Multi-Level Preference Evidence (MLPE)**: It uses only the semantic difference at the first divergence level $d_{i,j}$ and removes the level-wise aggregation across all divergence levels.
- **w/o Multi-Stage Injection (MSI)**: It removes the multi-stage evidence injection and injects preference evidence only once at the final layer of the LLM.

Table 2 reports the ablation results. Removing the Preference Evidence Miner (w/o PEM) causes a substantial performance drop on both tasks, confirming that modeling preference evidence from co-exposed candidates is crucial for user decision modeling. Removing SID Knowledge Injection (w/o KI) also leads to a substantial decrease, indicating that aligning SID tokens with textual semantics is essential for the LLM to interpret item content. Replacing structured evidence with an MLP (w/o SPE) reduces performance under the same injection mechanism, demonstrating the value of hierarchical difference modeling. Removing Multi-Level Preference Evidence (w/o MLPE) or Multi-Stage Injection (w/o MSI) also degrades performance. The MLPE variant misses complementary decision cues across semantic levels, while the MSI variant injects evidence only at the final layer, limiting the LLM's ability to progressively fuse preference evidence. Appendix C.4 provides further ablation results.

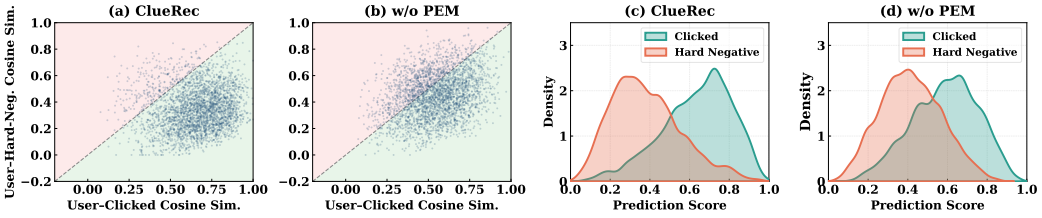

Figure 3: Preference evidence analysis. (a) Cosine similarity between the user representation and clicked items versus hard negatives for ClueRec, and (b) for w/o PEM. (c) Prediction score distributions for clicked items and hard negatives for ClueRec, and (d) for w/o PEM.

4.4 Preference Evidence Analysis

To further investigate preference evidence, we compare ClueRec with w/o PEM, which removes the entire preference evidence module and directly feeds clicked items and exposed candidates into the original LLM backbone. We randomly sample 3,000 users from the validation set, each contributing one impression event. For each event, we select the unclicked item most similar to the user representation from the exposure set as the hard negative. We plot the cosine similarities of the user representation to the clicked item and the hard negative in Figure 3-(a) and (b). If preference evidence is learned, the clicked item should have higher similarity, placing points below the diagonal (green region). Most ClueRec points fall in this region, whereas w/o PEM has more points in the red region, indicating that removing PEM hinders the user representation's ability to distinguish clicked items from hard negatives.

We further compare the prediction score distributions of clicked items and hard negatives (Figure 3-(c) and (d)). We use kernel density estimation (KDE) to visualize these distributions and assess their separation. ClueRec yields two well-separated KDE curves, with clicked scores concentrated at higher values and hard-negative scores at lower values, whereas w/o PEM shows heavier overlap. This indicates that preference evidence is not only encoded in the representation space but also propagated to the final predictions. These analyses show that ClueRec learns not only what users click but also why they prefer it over co-exposed alternatives.

4.5 Hyperparameter Analysis

We analyze two hyperparameters central to ClueRec: the number of SID levels L and the number of preference-evidence injections T. We vary one parameter at a time while keeping the other settings fixed. As shown in Figure 4, the trends for both GR and CTR improve as L increases to the default depth, then decline slightly. A deeper hierarchy may capture finer differences between clicked

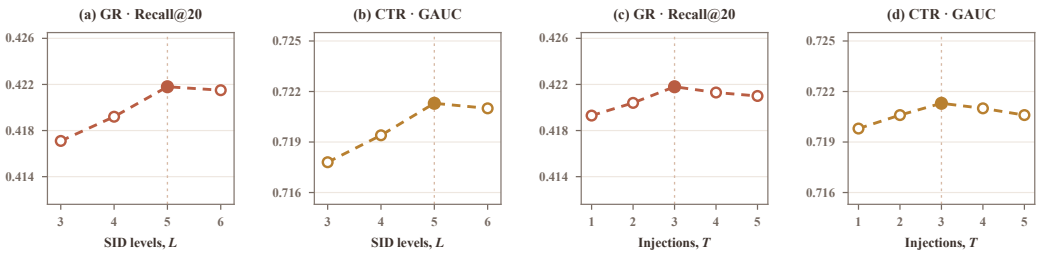

Figure 4: Hyperparameter analysis on SID levels L and the number of preference-evidence injections T.

items and their co-exposed alternatives, whereas additional levels may introduce redundant detail. Increasing T also improves performance up to the default setting, suggesting that successive injections help the model integrate preference evidence. Further injections show no additional benefit and may repeatedly introduce the same clues.

4.6 Case Study

Figure 5 shows two clicked items from the same user. In the upper part, each clicked item and its co-exposed candidates form a SID tree. The orange dashed boxes highlight key divergence points between the clicked path and the exposure branches. In both cases, the main divergences occur at levels c and d. The lower-left panel shows sorted gating scores, where the top-4 entries correspond to these marked branches, indicating that ClueRec focuses on these levels. The lower-right panel compares prediction scores with and without the four key evidence. With these key evidences, the clicked item (green) scores clearly higher than all exposed candidates (gray). Without them, the clicked score becomes close to or even lower than some candidates. This shows that the extracted evidence is important for distinguishing the clicked item from its alternatives.

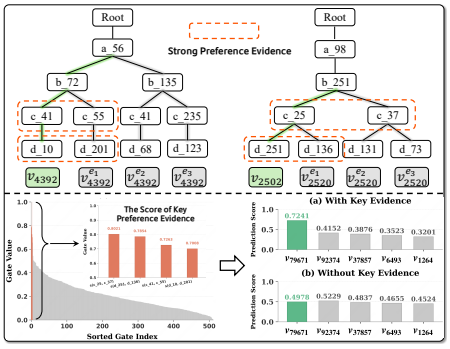

Figure 5: Case Study of ClueRec.

4.7 Online A/B Testing

We deploy ClueRec as a user embedding provider to boost the online production model in the advertising recommendation scenario of a large-scale platform serving hundreds of millions of users. The online A/B test runs for one week and covers 20% of advertising traffic. Compared with the production baseline, ClueRec improves CTR by 0.571% and revenue per mille (RPM) by 1.024% on average, both significant under our production criterion. Both metrics consistently outperform the baseline throughout the seven-day experiment, demonstrating ClueRec's stable effectiveness in improving user engagement and platform revenue.

5 Conclusion

In this work, we identify the omission of preference evidence in existing LLM-based recommendation models, which primarily learn from positive interactions while overlooking co-exposed alternatives. We propose ClueRec, which organizes clicked and co-exposed items into hierarchical SID trees, extracts coarse-to-fine preference evidence from divergent branches, and progressively integrates it into LLM-based user representations. Evaluated separately on generative recommendation and CTR prediction, ClueRec consistently outperforms strong baselines, with ablations and further analyses validating its key components. A one-week online A/B test also shows stable gains in CTR and RPM, demonstrating the value of within-exposure comparisons for practical recommendation.

LLM USAGE DISCLOSURE

During the preparation of this manuscript, GPT-5.6 was used exclusively for grammar correction, wording refinement, and readability improvement. It was not used to develop the research ideas, methodology, experiments, analysis, or conclusions. All AI-assisted suggestions were independently reviewed and approved by the authors, who retain full responsibility for the content and integrity of the manuscript.

ETHICS STATEMENT

We report only aggregate statistics and results and do not disclose individual-level user records or personally identifiable information. Due to privacy and proprietary restrictions, we do not publicly release industrial logs, sample records, or online A/B-test data.

REPRODUCIBILITY STATEMENT

An anonymous repository at https://anonymous.4open.science/r/ClueRec-2E8B provides the implementation, configuration files, and preprocessing and evaluation scripts for ClueRec. The data construction and chronological splitting procedures are described in Appendix C.1, with implementation details and hyperparameters provided in Appendix B.2. Due to privacy and proprietary constraints, we do not release the industrial dataset, sample records, or online A/B-test logs. The code supports implementation and evaluation on compatible data, while reproduction of the reported results requires access to the internal platform data.

REFERENCES

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

Yu Cui, Feng Liu, Jiawei Chen, Xingyu Lou, Changwang Zhang, Jun Wang, Yuegang Sun, Xiaohu Yang, and Can Wang. Field matters: A lightweight llm-enhanced method for ctr prediction. In *Proceedings of the ACM Web Conference 2026*, pp. 6365–6376, 2026.

Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.

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

Zhifang Fan, Dan Ou, Yulong Gu, Bairan Fu, Xiang Li, Wentian Bao, Xin-Yu Dai, Xiaoyi Zeng, Tao Zhuang, and Qingwen Liu. Modeling users' contextualized page-wise feedback for click-through rate prediction in e-commerce search. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pp. 262–270, 2022.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, pp. 299–315, 2022.

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

Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, et al. Plum: Adapting pre-trained language models for industrial-scale generative recommendations. In *Proceedings of the ACM Web Conference 2026*, pp. 8093–8104, 2026.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 585–593, 2022.

Yupeng Hou, Jiacheng Li, Ashley Shin, Jinsung Jeon, Abhishek Santhanam, Wei Shao, Kaveh Hassani, Ning Yao, and Julian McAuley. Generating long semantic ids in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 956–966, 2025.

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

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Keqin Chen, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, et al. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026.

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

Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. Modeling user exposure in recommendation. In *Proceedings of the 25th international conference on World Wide Web*, pp. 951–961, 2016.

Yu Liang, Zhongjin Zhang, Yuxuan Zhu, Kerui Zhang, Zhiluohan Guo, Wenhang Zhou, Zonqi Yang, Kangle Wu, Yabo Ni, Anxiang Zeng, et al. Rethinking generative recommender tokenizer: Recsys-native encoding and semantic quantization beyond llms. *arXiv preprint arXiv:2602.02338*, 2026.

Xinyu Lin, Chaoqun Yang, Wenjie Wang, Yongqi Li, Cunxiao Du, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. Efficient inference for large language model-based generative recommendation. In *International Conference on Learning Representations*, volume 2025, pp. 91672–91697, 2025.

Xinyu Lin, Pengyuan Liu, Wenjie Wang, Yicheng Hu, Chen Xu, Fuli Feng, Qifan Wang, and Tat-Seng Chua. Bringing reasoning to generative recommendation through the lens of cascaded ranking. In *Proceedings of the ACM Web Conference 2026*, pp. 8939–8949, 2026.

Zhanyu Liu, Shiyao Wang, Xingmei Wang, Rongzhou Zhang, Jiaxin Deng, Honghui Bao, Jinghao Zhang, Wuchao Li, Pengfei Zheng, Xiangyu Wu, et al. Onerec-think: In-text reasoning for generative recommendation. *arXiv preprint arXiv:2510.11639*, 2025.

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.

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

Zihua Si, Lin Guan, ZhongXiang Sun, Xiaoxue Zang, Jing Lu, Yiqun Hui, Xingchao Cao, Zeyu Yang, Yichen Zheng, Dewei Leng, et al. Twin v2: Scaling ultra-long user behavior sequence modeling for enhanced ctr prediction at kuaishou. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pp. 4890–4897, 2024.

OneRec Team, Biao Yang, Boyang Ding, Chenglong Chu, Dunju Zang, Fei Pan, Han Li, Hao Jiang, Honghui Bao, Huanjie Wang, et al. Onereason technical report. *arXiv preprint arXiv:2606.06260*, 2026.

Qwen Team. Qwen3. 5-omni technical report. *arXiv preprint arXiv:2604.15804*, 2026.

Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pp. 1652–1656, 2021.

Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Daifeng Guo, Yanli Zhao, Shen Li, Yuchen Hao, Yantao Yao, et al. Wukong: Towards a scaling law for large-scale recommendation. *arXiv preprint arXiv:2403.02545*, 2024.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 37(5):2329–2340, 2025.

Ziqi Zhao, Zhaochun Ren, Jiyuan Yang, Zuming Yan, Zihan Wang, Liu Yang, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Xin Xin. Improving sequential recommenders through counterfactual augmentation of system exposure. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1508–1518, 2025.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 1435–1448. IEEE, 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.

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

Guorui Zhou, Honghui Bao, Jiaming Huang, Jiaxin Deng, Jinghao Zhang, Junda She, Kuo Cai, Lejian Ren, Lu Ren, Qiang Luo, et al. Openonerec technical report. *arXiv preprint arXiv:2512.24762*, 2025.

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

Appendix

- **A. Related Work** 13
- **B. Additional Methodology** 14
  - B.1 *Theoretical Analysis of Preference Evidence Learning* 14
  - B.2 *Algorithm Flow and Pseudocode* 16
  - B.3 *Parameters Setting and Implementation Details* 17
- **C. Supplementary Experiments** 17
  - C.1 *Training Data Composition and Mixing Ratios* 17
  - C.2 *Evaluation Metrics* 18
  - C.3 *Baseline Introduction* 18
  - C.4 *Additional Ablation Study* 19
  - C.5 *Complexity and Inference Efficiency* 20
  - C.6 *Choice of Knowledge Injection Objective* 20

A Related Work

Generative Recommendation. Generative recommendation originated from a shift in item representation. Early recommender systems (Koren et al., 2009; He et al., 2017) used random IDs as embedding table indices, which lack semantic information and generalize poorly to cold-start and long-tail items. UniSRec (Hou et al., 2022) and P5 (Geng et al., 2022) unified pretrained language models with recommendation, leveraging textual semantics to enhance item representation and user sequence modeling. Later, residual quantization methods such as RQ-VAE converted item content embeddings into hierarchical Semantic IDs (SIDs), enabling item identifiers to carry coarse-to-fine semantics. TIGER (Rajput et al., 2023) first introduced SIDs into generative retrieval, autoregressively generating the target item's SID with an encoder-decoder. RPG (Hou et al., 2025) produces unordered long SIDs and predicts all tokens in parallel, improving both expressiveness and inference efficiency. CARE (Lin et al., 2026) views autoregressive SID generation as cascaded ranking and introduces progressive history encoding and query-anchored reasoning to mitigate bias amplification across generation steps. ReSID (Liang et al., 2026) instead advocates a recommendation-native SID framework that quantizes predictive item representations for information preservation and sequential predictability.

With the rise of LLMs, SID-based methods has been further integrated with LLMs. OneRec (Deng et al., 2025) reframes recommendation as LLM-based end-to-end autoregressive generation. At-Speed (Lin et al., 2025) addresses the inference latency of this paradigm with a top-K alignment framework for speculative decoding. Subsequently, a series of works focus more on enabling LLMs to understand SIDs. OneRec-Think (Liu et al., 2025) proposes Itemic Alignment, which projects itemic tokens into the LLM's textual space and activates the LLM's reasoning ability. OpenOneRec (Zhou et al., 2025) open-sources this line as a Qwen3-based model series, injecting collaborative signals through Itemic-Text Alignment and full-parameter collaborative pretraining. OneReason (Team et al., 2026) builds itemic token perception through four-granularity itemic-text alignment data during pretraining and constructs chain-of-thought (CoT) capabilities. Another line designs dedicated alignment tasks or adaptation pipelines. LC-Rec (Zheng et al., 2024) introduces a learning-based vector quantization method to ground SIDs in language and collaborative semantics; PLUM (He et al., 2026) proposes a multi-stage adaptation recipe that transfers LLMs to YouTube-scale industrial recommendation via SID tokenization, continual pretraining, and task fine-tuning.

Sequence Modeling for CTR Prediction. CTR sequence modeling has evolved from feature interaction (Cheng et al., 2016; Lian et al., 2018), to interest modeling (Zhou et al., 2018; 2019), and further to long-sequence retrieval and unified Transformers (Zhu et al., 2025; Si et al., 2024). Early CTR models, represented by DeepFM (Guo et al., 2017), primarily capture low- and high-order feature interactions, with user histories often simply pooled. DIN (Zhou et al., 2018) first introduced attention into CTR, learning user interest representations from historical behaviors adaptively to the candidate item. DIEN (Zhou et al., 2019) further modeled how interests evolve over time. As user behavior sequences grew longer, SIM (Pi et al., 2020) proposed a two-stage cascaded retrieval, first retrieving candidate-relevant behaviors from ultra-long sequences. More recently, WuKong (Zhang et al.,

2024) scales feature interaction modeling through stacked factorization and compression blocks. LLMs have also been introduced to enrich CTR prediction. CoLLM (Zhang et al., 2025) maps collaborative user and item representations into the LLM embedding space for recommendation, whereas LLaCTR (Cui et al., 2026) distills field-level semantics from LLMs to enhance the feature representations and interactions of conventional CTR models. Beyond sequence modeling itself, the use of exposure information provides another relevant line of work. ExpoMF (Liang et al., 2016) models exposure as a latent variable, while RACP (Fan et al., 2022) captures intra-page exposure context and cross-page interest evolution for CTR prediction. CaseRec (Zhao et al., 2025) takes exposure sequences as input and models them with a decision Transformer.

B Additional Methodology

B.1 Theoretical Analysis of Preference Evidence Learning

This subsection analyzes how clicked–exposed comparisons help learn user preferences and how the Preference Evidence Miner makes better use of exposure information. We first show that multi-level semantic differences between clicked and co-exposed items can reflect relative preferences. We then separate exposure comparisons into stable preference signals and random noise, and analyze how ClueRec preserves the former while reducing the latter through selective aggregation.

Consider an impression with a clicked item v^+, co-exposed alternatives $\mathcal{E}$, and the complete choice set $\mathcal{C} = \{v^+\} \cup \mathcal{E}$. Let $\boldsymbol{\theta}$ represent the user's preference direction and $\boldsymbol{\phi}_{\mathrm{pref}}(v)$ denote the item's idealized preference-relevant features. We describe the user's choice using a conditional logit model:

$$p_{\boldsymbol{\theta}}(v \mid \mathcal{C}) = \frac{\exp\big(\boldsymbol{\theta}^\top \boldsymbol{\phi}_{\mathrm{pref}}(v)\big)}{\sum_{w \in \mathcal{C}} \exp(\boldsymbol{\theta}^\top \boldsymbol{\phi}_{\mathrm{pref}}(w))}. \tag{18}$$

Here, $\boldsymbol{\theta}^\top \boldsymbol{\phi}_{\mathrm{pref}}(v)$ is the utility of item v for the user. For the clicked item v^+ and a co-exposed item v^e,

$$\log \frac{p_{\boldsymbol{\theta}}(v^+ \mid \mathcal{C})}{p_{\boldsymbol{\theta}}(v^e \mid \mathcal{C})} = \boldsymbol{\theta}^\top \left[\boldsymbol{\phi}_{\mathrm{pref}}(v^+) - \boldsymbol{\phi}_{\mathrm{pref}}(v^e)\right]. \tag{19}$$

This identity shows that relative preference can be described through differences between the items' preference-relevant features. Co-exposed items provide the alternatives against which a click is interpreted, allowing the model to understand the choice through relative features.

To analyze the SID hierarchy, assume that these idealized features have an additive form:

$$\boldsymbol{\phi}_{\mathrm{pref}}(v) = \sum_{l=1}^{L} \mathbf{e}_{\mathrm{pref}, z_v^l}, \tag{20}$$

where $\mathbf{e}_{\mathrm{pref}, z_v^l}$ is the ideal preference feature associated with the SID token at level l. Consider a clicked–exposed pair with distinct SIDs, and let d be its first divergence level. Since the prefixes before d are identical, their contributions cancel:

$$\boldsymbol{\phi}_{\mathrm{pref}}(v^+) - \boldsymbol{\phi}_{\mathrm{pref}}(v^e) = \sum_{l=d}^{L} \boldsymbol{\delta}_{\mathrm{pref}}^l, \text{ where } \boldsymbol{\delta}_{\mathrm{pref}}^l = \mathbf{e}_{\mathrm{pref}, z^{+,l}} - \mathbf{e}_{\mathrm{pref}, z^{e,l}}, \tag{21}$$

Based on Eq. (19), we can drive:

$$\boxed{\log \frac{p_\theta(v^+ \mid \mathcal{C})}{p_\theta(v^e \mid \mathcal{C})} = \sum_{l=d}^{L} \boldsymbol{\theta}^\top \boldsymbol{\delta}_{\mathrm{pref}}^l}. \tag{22}$$

This result shows that relative preference may depend on several levels after the first divergence. The first divergence identifies where the items begin to differ, but later levels may still contain information that affects the choice. ClueRec therefore extracts multi-level differences from the first divergence onward to capture preference evidence at different semantic granularities.

The observed clicked–exposed difference at level l is

$$\boldsymbol{\delta}^l = \mathbf{e}_{z^{+,l}} - \mathbf{e}_{z^{e,l}}. \tag{23}$$

However, not all semantic differences are relevant to the current user's preferences. Some provide useful preference evidence, while others may interfere with its extraction. We therefore decompose the observed difference as

$$\boldsymbol{\delta}^l = \boldsymbol{\delta}^l_{\text{pref}} + \boldsymbol{\epsilon}^l, \tag{24}$$

where $\boldsymbol{\epsilon}^l$ contains differences unrelated to the user's preferences, referred to as comparison noise. The Preference Evidence Miner aims to preserve preference signals while reducing the effect of this noise.

We analyze historical clicked–exposed pairs from the same user with the same first divergence level. Here, $\mathbb{E}$ denotes the average over these comparisons. We assume that preference signals and comparison noise are orthogonal on average, i.e., $\mathbb{E}\left[\left\langle \boldsymbol{\delta}^l_{\text{pref}}, \boldsymbol{\epsilon}^l \right\rangle\right] = 0$.

We define the average signal and noise strengths at level l as

$$S_l = \mathbb{E}\|\boldsymbol{\delta}^l_{\text{pref}}\|_2^2, \qquad N_l = \mathbb{E}\|\boldsymbol{\epsilon}^l\|_2^2. \tag{25}$$

Both are scalars measuring the magnitudes of the preference signal and noise. We consider levels containing both signal and noise, with $0 < S_l < \infty$ and $0 < N_l < \infty$.

First, consider a fixed weight $w \in [0, 1]$. The weighted difference is $w\boldsymbol{\delta}^l$, and its mean squared error relative to the ideal preference difference is

$$\mathcal{R}_l(w) = \mathbb{E}\left\|w\boldsymbol{\delta}^l - \boldsymbol{\delta}^l_{\text{pref}}\right\|_2^2. \tag{26}$$

Substituting the signal–noise decomposition gives

$$w\boldsymbol{\delta}^l - \boldsymbol{\delta}^l_{\text{pref}} = (w-1)\boldsymbol{\delta}^l_{\text{pref}} + w\boldsymbol{\epsilon}^l. \tag{27}$$

Based on Eqs. (25) and (26), we expand the squared norm and apply the average orthogonality assumption, which yields

$$\mathcal{R}^l(w) = (1-w)^2 S_l + w^2 N_l. \tag{28}$$

This equation captures two effects of level selection. Reducing the weight weakens part of the preference signal, increasing the first error term. It also reduces the retained noise, decreasing the second term. An appropriate weight therefore balances signal loss and noise reduction.

Minimizing this error over w gives the optimal fixed weight $w_l^* = \frac{S_l}{S_l + N_l}$. When the preference signal is relatively strong, w_l^* approaches 1, so more of the level-wise difference is retained. When noise is relatively strong, w_l^* decreases, so the difference is more strongly suppressed.

With this optimal weight, the mean squared error in estimating the ideal preference difference becomes

$$\mathcal{R}_l(w_l^*) = \frac{S_l N_l}{S_l + N_l} < N_l = \mathcal{R}_l(1). \tag{29}$$

Here, $w = 1$ corresponds to using the original difference without weighting. Thus, suitable weighting can make the benefit of noise reduction exceed the cost of signal attenuation, producing more accurate preference evidence than the original difference.

The error can also be written as

$$\mathcal{R}_l(w) = \frac{S_l N_l}{S_l + N_l} + (S_l + N_l)(w - w_l^*)^2. \tag{30}$$

A larger deviation from w_l^* therefore produces a larger additional error. Since signal and noise strengths can vary across levels, different levels may require different weights. This motivates ClueRec's use of separate gates for different levels.

ClueRec computes gate score g^l from the shared prefix, the current level-wise difference, and the level embedding, so the gate varies with the input. To analyze the learned gate, define the root mean squared distance η_l between its output and the optimally weighted output:

$$\eta_l = \left(\mathbb{E}\left\|g^l\boldsymbol{\delta}^l - w_l^*\boldsymbol{\delta}^l\right\|_2^2\right)^{1/2}. \tag{31}$$

In addition, decomposing the learned gate's estimation error gives

$$g^l \boldsymbol{\delta}^l - \boldsymbol{\delta}^l_{\text{pref}} = \left(w_l^* \boldsymbol{\delta}^l - \boldsymbol{\delta}^l_{\text{pref}}\right) + (g^l - w_l^*)\boldsymbol{\delta}^l. \tag{32}$$

Using Eqs. (31)–(32), we apply the L_2 triangle inequality, which yields

$$\boxed{\left(\mathbb{E}||g^l \boldsymbol{\delta}^l - \boldsymbol{\delta}^l_{\text{pref}}||_2^2\right)^{1/2} \leq \sqrt{\frac{S_l N_l}{S_l + N_l}} + \eta_l}. \tag{33}$$

Therefore, if

$$\eta_l < \sqrt{N_l} - \sqrt{\frac{S_l N_l}{S_l + N_l}}, \tag{34}$$

we can obtain

$$\boxed{\mathbb{E}\left\|g^l \boldsymbol{\delta}^l - \boldsymbol{\delta}^l_{\text{pref}}\right\|_2^2 < \mathbb{E}\left\|\boldsymbol{\delta}^l - \boldsymbol{\delta}^l_{\text{pref}}\right\|_2^2}. \tag{35}$$

These results provide a theoretical basis for ClueRec's Preference Evidence Miner. Multi-level clicked–exposed differences capture preference contributions across SID levels, while selective weighting can reduce comparison noise. Under the stated assumptions, a learned gate achieves lower estimation error than the unweighted difference when its output is sufficiently close to the optimally weighted output. This gives a sufficient condition under which ClueRec's level-wise gating produces more accurate preference evidence for user preference modeling.

B.2 Algorithm Flow and Pseudocode

Algorithm 1 summarizes the workflow of ClueRec. First, the Semantic Mapper combines item text and image features, constructs five-level SIDs with RQ-VAE, and uses SID-to-text reconstruction to learn the meanings of the new SID tokens. For each recommendation minibatch, the Preference Evidence Miner compares the SID paths of clicked items and their co-exposed alternatives. It extracts semantic differences from their first divergence onward, builds an evidence memory, and injects this evidence into the user representation through attention. ClueRec is instantiated and trained separately for generative recommendation and CTR prediction. Training also includes general-domain QA and SID knowledge-injection data to preserve the LLM's general capabilities and its understanding of SID semantics.

Table 3: Parameter settings and implementation details of ClueRec.

Component	**Parameter**	**Value**
Implementation	Framework	PyTorch 2.10; HuggingFace Transformers
	LLM backbone	Qwen3.5-0.8B
	Precision	bf16
Semantic Mapper	Text/image encoder	Qwen3-VL (8B)
	SID levels L	5
	Codebook size	$[512, 512, 512, 512, 512]$
	Knowledge Injection Task	SID-to-text reconstruction
	Frozen Parameters	Original Qwen parameters
	Trainable Parameters	Newly added SID token embeddings
	SID Token Count	2560
	Learning Rate / Batch Size / Epochs	2e-5 / 1024 / 10
Preference Evidence Miner	Max History N	128
	Max Co-exposed items M	4
	Evidence Memory Size P_u	512
	Injection Position	Before the full-attention layer in each Qwen stage
	Number of Interactions T	3
	Positional Encoding / Causal Mask	No / No
Recommender	Shared Modules	LLM backbone and SID vocabulary
	GR Decoding	Constrained prefix-tree decoding
	GR Beam Size	100
	CTR Head	MLP $[512, 256, 1]$
	Learning Rate / Batch Size / Epochs	5e-4 / 128 / 8

Algorithm 1 Implementation of ClueRec

Input: Items $\mathcal{I}$, pre-target clicked histories $\{\mathcal{H}_u\}$, and training data.
Output: Separate GR and CTR models.

 # Semantic Mapper
Construct SIDs via RQ-VAE (Eq. (1)).
Expand the vocabulary and perform knowledge injection (Eqs. (2)–(3)).

 # Preference Evidence Miner
function BUILDEVIDENCE($\mathcal{H}_u$)
  Compute shared features and gated differences (Eqs. (4)–(8)).
  return $\mathbf{M}_u$ (Eq. (9)).
end function

 # Recommender Training
for each task $\tau \in \{\mathrm{GR}, \mathrm{CTR}\}$ separately **do**
  for each minibatch from the recommendation/QA/SID-to-text mixture **do**
    if recommendation minibatch **then**
      Call BUILDEVIDENCE($\mathcal{H}_u$) for each example.
      Run the task forward pass; inject nonempty $\mathbf{M}_u$ into $\langle \mathrm{USER} \rangle$ (Eqs. (10)–(15)).
      $\mathcal{L} \leftarrow \mathcal{L}_\tau$ (Eqs. (14) and (17)).
    else
      Compute the autoregressive QA or SID-to-text loss $\mathcal{L}$.
    end if
    Backpropagate $\mathcal{L}$ and update trainable parameters.
  end for
end for

B.3 Parameters Setting and Implementation Details

We implement ClueRec in PyTorch 2.10 with HuggingFace Transformers. The LLM backbone is Qwen3.5-0.8B. Table 3 summarizes the main hyperparameters. For the Semantic Mapper, we use Qwen3-VL to encoder text and image and fuse their features by element-wise addition. RQ-VAE constructs five-level SIDs with codebook size $[512, 512, 512, 512, 512]$. In the SID knowledge injection stage, we freeze the original Qwen parameters and update only the newly added SID token embeddings using the SID-to-text reconstruction objective.

For the Preference Evidence Miner, each user history contains at most $N = 128$ impression events, and each event contains at most $M = 4$ co-exposed candidates. The evidence memory is truncated or sampled to at most $P_u = 512$ entries. We inject preference evidence before the full-attention layer in each Qwen stage, for $T = 3$ interactions.

For the Multi-Task Recommender, GR and CTR are trained separately but share the same LLM backbone and SID vocabulary. GR uses constrained prefix-tree decoding with beam size 100. The CTR head is an MLP with hidden sizes $[512, 256, 1]$ and dropout.

C Supplementary Experiments

C.1 Training Data Composition and Mixing Ratios

We construct the offline dataset by sampling 5 million users from 7 consecutive days of exposure and click logs, retaining both clicked and zero-click impressions. For each user, impressions are ordered chronologically: the last is used for testing, the penultimate for validation, and the remainder for training. Both tasks share this dataset and split. CTR evaluation includes zero-click impressions, whereas GR evaluation excludes them because no clicked target is available. When predicting a target impression, only prior user history is used; its click outcomes and co-exposed candidates are excluded from the preference evidence. Each user history contains at most 512 impression events, with up to four co-exposed alternatives per event. For generative recommendation (GR), the target is the five-level Semantic

ID (SID) of the clicked item in the target impression. For click-through rate (CTR) prediction, we use the candidates actually shown in that impression and their logged click labels.

As shown in Figure 6, recommendation data account for 70% of training mini-batches in each of the separately trained GR and CTR models. NQ-Open, TriviaQA, HotpotQA, and SQuAD 2.0 each account for 5%, while SID-to-text data, comprising five-level item SIDs paired with their textual descriptions, account for the remaining 10%. We use only the training splits of the four QA datasets, which provide auxiliary supervision for factual question answering, multi-hop reasoning, and reading comprehension.

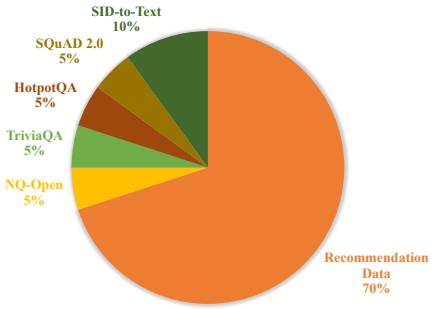

Figure 6: Training data composition for ClueRec.

C.2 Evaluation Metrics

Generative recommendation. We evaluate each user on their final impression. The model uses constrained decoding to produce a ranked list of valid item SIDs. Let r_u be the rank of user u's clicked item, with $r_u = \infty$ if the item is not retrieved. For $K \in \{20, 50\}$, we compute

$$\text{Recall@}K = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbf{1}(r_u \le K), \qquad \text{NDCG@}K = \frac{1}{|\mathcal{U}|} \sum_{u : r_u \le K} \frac{1}{\log_2(r_u + 1)}. \tag{36}$$

For GR evaluation, each test example has one target clicked item, so Recall@K is equivalent to Hit Rate@K. NDCG@K also accounts for its rank.

CTR prediction. We evaluate the predicted click probabilities of items actually shown in the test impressions. AUC measures how often a clicked item receives a higher score than an unclicked item. GAUC averages AUC across users, weighted by the number of test candidates n_u:

$$\text{GAUC} = \frac{\sum_{u \in \mathcal{U}^*} n_u \, \text{AUC}_u}{\sum_{u \in \mathcal{U}^*} n_u}, \tag{37}$$

where $\mathcal{U}^*$ contains users with both clicked and unclicked candidates. LogLoss measures the accuracy of the predicted probabilities:

$$\text{LogLoss} = -\frac{1}{J} \sum_{j=1}^{J} \left[y_j \log \hat{p}_j + (1 - y_j) \log(1 - \hat{p}_j) \right]. \tag{38}$$

Higher Recall, NDCG, AUC, and GAUC are better; lower LogLoss is better. We repeat each experiment five times and report the mean.

C.3 Baseline Descriptions

We compare ClueRec with representative baselines for generative recommendation and CTR prediction. The GR baselines span conventional sequential recommendation, SID-based generation, exposure-aware sequence modeling, and reasoning-enhanced generation. The CTR baselines cover feature interaction, behavior sequence modeling, exposure-aware modeling, and LLM-enhanced prediction.

Generative Recommendation Baselines.

- **SASRec** (Kang & McAuley, 2018) uses causal self-attention to capture sequential dependencies in user interactions for next-item prediction.
- **TIGER** (Rajput et al., 2023) quantizes item semantics into hierarchical SIDs and uses an encoder–decoder model to autoregressively generate the SID of the next item.
- **LC-Rec** (Zheng et al., 2024) constructs non-conflicting item indices through vector quantization and uniform semantic mapping, and aligns them with language and collaborative semantics through instruction tuning.

- **CaseRec** (Zhao et al., 2025) models exposure sequences with a Decision Transformer and uses counterfactual augmentation to explore potential user interests.
- **OpenOneRec** (Zhou et al., 2025) represents items as hierarchical itemic tokens and adapts a Qwen3 backbone through itemic–text alignment, collaborative co-pretraining, and recommendation post-training.
- **CARE** (Lin et al., 2026) augments TIGER with progressive history encoding and query-anchored reasoning to mitigate bias amplification during SID generation. We keep the SID tokenizer and generative backbone identical to those of the standalone TIGER baseline to isolate the contribution of CARE.

CTR Prediction Baselines.

- **DeepFM** (Guo et al., 2017) combines a factorization machine with a deep neural network to learn low- and high-order feature interactions.
- **DIEN** (Zhou et al., 2019) extracts users' sequential interests and models their evolution with respect to the candidate item.
- **SIM** (Pi et al., 2020) retrieves candidate-relevant behaviors from long user histories and models their relation to the candidate item.
- **RACP** (Fan et al., 2022) uses page-context-aware and recurrent attention to capture within-page comparisons and cross-page interest evolution from exposure–feedback sequences.
- **CoLLM** (Zhang et al., 2025) maps the user and item embeddings produced by DIN into the LLM token space through its collaborative information encoding module. We use DIN as the collaborative encoder and denote this instantiation CoLLM-DIN.
- **LLaCTR** (Cui et al., 2026) applies LLaCTR to the WuKong (**?**) backbone. LLM-derived field semantics guide its feature representations, while an auxiliary field-interaction network produces a logit that is fused with the original WuKong prediction.

We use the official implementations where available and tune all baselines on the same validation split. All methods follow the same chronological training, validation, and test split. GR methods predict the held-out clicked item from preceding interactions, whereas CTR methods score the same logged exposure candidates against their click labels.

Table 4: Additional ablation study of ClueRec. All experiments are repeated five times, and the average results are reported. † denotes statistical significance under a paired t-test ($p < 0.05$).

Methods	Generative Recommendation				CTR Prediction		
	Recall@20↑	NDCG@20↑	Recall@50↑	NDCG@50↑	AUC↑	GAUC↑	LogLoss↓
w/o LG	0.4186	0.2596	0.5350	0.2827	0.7532	0.7196	0.3815
FD	0.4162	0.2547	0.5323	0.2802	0.7514	0.7178	0.3825
ClueRec	**0.4218**†	**0.2612**†	**0.5379**†	**0.2847**†	**0.7545**†	**0.7213**†	**0.3807**†

C.4 Additional Ablation Study

To further validate ClueRec, we conduct two additional ablations:

- **w/o Level-wise Gating (LG)**: It sets all gates to one, uniformly averaging differences from the first divergence onward while retaining shared-prefix encoding.
- **Flat Difference (FD)**: It replaces structured evidence with the difference between mean-pooled clicked and co-exposed SID embeddings, removing shared-prefix encoding and level-wise gating.

Both variants retain the same evidence injection mechanism and are retrained under the same settings as ClueRec.

Table 4 shows that removing level-wise gating degrades performance on both tasks, supporting the benefit of selectively weighting semantic differences. Flat Difference also underperforms ClueRec, indicating that simple item-level subtraction is less effective than the proposed structured evidence modeling.

C.5 Complexity and Inference Efficiency

To examine the effect of model size on ClueRec, we compare Qwen3.5-0.8B, 2B, and 4B using the same eight A100 GPUs. We measure recommendation performance, training time, inference latency, and GPU memory. As shown in Table 5, larger models improve both GR and CTR performance but increase computational cost. At each model size, CTR is less costly because it reuses the user representation and directly scores candidate items, whereas GR generates a five-level SID. We therefore use Qwen3.5-0.8B as the default backbone to balance recommendation quality with the latency and memory requirements of online deployment.

Table 5: Efficiency and performance of ClueRec across Qwen3.5 model sizes. GR/CTR values are listed in order; memory is summed over eight A100 GPUs.

LLM Size	Training GR/CTR		Inference GR/CTR		Performance	
	Time/Epoch (h)	GPU Costs (GB)	Time/Sample (s)	GPU Costs (GB)	Recall@20 (GR)	GAUC (CTR)
Qwen3.5-0.8B	2.28/2.08	214.08/203.14	0.18/0.11	96.54/85.46	0.4218	0.7213
Qwen3.5-2B	4.56/4.17	306.44/286.56	0.32/0.19	106.78/92.73	0.4256	0.7236
Qwen3.5-4B	6.67/6.14	577.34/526.32	0.65/0.29	201.58/188.92	0.4289	0.7252

C.6 Choice of Knowledge Injection Objective

In early experiments, we tried text-to-SID generation and other knowledge injection objectives, but the new SID token embeddings quickly exploded in norm and made training unstable. We attribute this to the text-to-SID setup. SID tokens are absent from Qwen's vocabulary and have no natural-language prior, and five-level SID combinations are long-tailed. When SID tokens are used as output targets, the model must perform a softmax over the extended SID vocabulary. Supervision for the correct SID token is sparse and sharp, so low-frequency tokens are under-updated while high-frequency tokens are over-updated. If input and output embeddings are tied, output gradients also flow back into the SID input embeddings and amplify their norms. SID-to-text avoids these problems. It generates item text conditioned on SIDs, keeping the output space in Qwen's original vocabulary. We freeze the original Qwen parameters and update only the input embeddings of the new SID tokens. The SID embeddings receive smooth semantic gradients from dense text descriptions such as categories, brands, and attributes, and they do not compete in the output layer. Training is thus more stable.