IMPROVE DISCRIMINATIVE RECOMMENDATION THROUGH BETTER GENERATIVE PRETRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Generative next-item prediction (NTP) pretraining improves click-through rate (CTR) prediction even when only the item embeddings are transferred. This demonstrates that pretraining yields effective item representations. We decompose embedding gradients and find that positive-target supervision is mostly important. Specifically, we discover that item frequency and attention structure make great effects on gradients. Low item frequency causes the inefficient supervision on embedding while self-attention structure makes the gradient direction sensitive to the most recent interaction. We therefore propose two modifications of NTP pretraining, Frequency Adjustment (FA) and shared-query Cross attention (Cross). First, FA strengthens the output gradients for infrequent targets to mitigate the supervision imbalance. Second, Cross uses a learned shared query to aggregate user history, reducing the model's reliance on recent interactions. Both methods keep the next-item objective and the transfer pipeline. Experiments on public datasets show that joint method improves mean downstream AUC over standard NTP.

1 INTRODUCTION

Transferring and freezing only the item embeddings from sequential pretraining can improve click-through rate (CTR) prediction, even when the sequence encoder is discarded (Wang et al., 2025). This raises a design question for Next Item Prediction (NTP) pre-training:

How can the pre-training method be further improved to achieve better item embeddings and downstream performance?

To guide the design, we distinguish the sources of supervision that shape the item embeddings during NTP. In our setting, the same item embeddings are used for history inputs and prediction candidates, so each embedding can receive gradients as a historical input, an observed next-item target, or a sampled negative. We assess the downstream model's reliance on these sources by perturbing their accumulated contributions while keeping its other parameters fixed. We find that perturbing the positive-target contribution generally causes the largest drop in AUC, suggesting that this supervision plays an important role in the transferred representations. This observation motivate us to consider modifications to positive-target gradient, including update allocation method and hidden state construction.

We first find that NTP's transfer gains vary with item frequency. To characterize this variation, we group positive and negative candidate items by their occurrence counts as pretraining targets and compare NTP with direct CTR training using downstream pairwise loss. NTP yields smaller loss reductions when an infrequent positive is compared with a frequent negative than in the reverse case. Infrequent items also have fewer opportunities to receive positive-target updates, since this path is activated only when an item appears as the observed next target. Together with the downstream model's sensitivity to these updates, this pattern motivates strengthening positive-target supervision for infrequent items.

Our analysis of the positive-target gradient also highlights the importance of the hidden state used for next-item prediction, since this state shapes the gradient's direction. In our NTP setting, the most recent item's embedding directly initializes this state. However, recommendation sequences may amplify the effect of this dependence: consecutive interactions can be weakly related, so the latest item need not provide the most relevant context for the next target. Our perturbation experiments

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.

show that changing the most recent item produces a larger change in the hidden state than perturbing the immediately preceding item by the same amount. These observations motivate constructing the predictive state from the observed history without giving the latest item a privileged role in its initialization.

Guided by these findings, we introduce two complementary modifications to NTP pretraining. Frequency Adjustment (FA) scales the positive-target gradient based on target frequency, assigning greater weight to infrequent targets and strengthening their learning signal along this supervision path. Shared-query Cross Attention (Cross) constructs the predictive state using a learned shared query that attends to the causal history, including the most recent item. This formulation allows the predictive state to emerge from an aggregation of historical context, rather than initializing it from the most recent item's embedding. The two modifications can be combined while preserving the forward computation of the NTP loss and the downstream transfer setting with frozen item embeddings.

Experiments on public datasets demonstrate that FA and Cross improve downstream AUC over standard next-token prediction (NTP), highlighting their effectiveness in learning transferable item embeddings. Comparisons across encoder depths further show that both methods retain average AUC gains with deeper pretraining encoders, suggesting that their benefits extend beyond shallow architectures. Together, these results suggest that the allocation of supervision and the construction of historical context offer valuable opportunities to improve representation transfer across encoder depths. FA and Cross therefore provide promising approaches for strengthening item embeddings through changes to the learning formulation, with benefits reflected in downstream predictive performance.

2 Related Work

Generative pretraining. Next-token pretraining enables large language models to support diverse downstream tasks (Brown et al., 2020; Touvron et al., 2023). Recommendation pretraining learns transferable representations through self-supervised objectives (Zhou et al., 2020; Hou et al., 2022) and unifies tasks through text generation (Geng et al., 2022). For CTR prediction, prior work explores behavior prediction (Wu et al., 2020a) and masked feature prediction or replaced feature detection (Lin et al., 2023). Recent work further combines generative pretraining with discriminative ranking to improve recommendation accuracy and model scalability (Wang et al., 2025; 2026; Li et al., 2026). Our work focuses on improving the item embeddings learned by NTP for downstream CTR prediction.

Structure of embeddings. The structure and learning dynamics of embeddings have been widely studied (Mu & Viswanath, 2018; Yao et al., 2025; 2026). In language modeling, research on representation degeneration (Gao et al., 2019) and frequency–semantics entanglement (Gong et al., 2018) highlights how token frequency can shape embedding geometry. In recommendation, related studies examine the alignment and uniformity of user and item representations (Wang et al., 2022), alongside popularity bias and the challenges of learning from sparse long-tail interactions (Park & Tuzhilin, 2008; Abdollahpouri et al., 2017). AGG mitigates degeneration by attenuating selected non-target output gradients to rare token embeddings (Yu et al., 2022), whereas FA strengthens positive-target output gradients to infrequent item embeddings for downstream CTR prediction with frozen embeddings.

Advances in recommendation architectures. Earlier work improves CTR architectures through feature interaction (Guo et al., 2017; Wang et al., 2021) and user-interest modeling (Zhou et al., 2018; 2019). Recent work develops scalable interaction backbones, including Wukong (Zhang et al., 2024), RankMixer (Zhu et al., 2025), and TokenMixer-Large (Jiang et al., 2026). OneTrans (Zhang et al., 2026) and WHALE (Cai et al., 2026) further integrate behavior sequences and non-sequential features within unified architectures. Within attention-based architectures, query and residual designs influence how context enters predictive states (Yang et al., 2019; Kozaki et al., 2026). Cross investigates shared-query attention during NTP pretraining, focusing on the quality of the item embeddings transferred to downstream CTR prediction.

3 Understanding Transfer through Item Embeddings

3.1 Preliminary

Direct CTR training. For click-through rate prediction task, the model predicts a click label $y \in \{0, 1\}$ based on a chronological user history $S = (x_1, \ldots, x_L)$ and a candidate item x. Generally speaking, parameters can be divided into item embedding $E \in \mathbb{R}^{V \times d}$ and the other dense parameters ϕ. Thus, the probability predicted by the model is $\hat{y} = f_{E,\phi}(S, x)$. Starting from random initialization, direct training jointly optimizes embedding and other dense parameters with binary cross-entropy:

$$\mathcal{L}_{\text{disc}}(E, \phi) = -\frac{1}{|\mathcal{D}|} \sum_{((S,x),y)\in\mathcal{D}} [y \log \hat{y} + (1 - y) \log(1 - \hat{y})] .$$

Next-item prediction pretraining. For next item prediction, the model encodes the prefix $S_{\le t} = (x_1, \ldots, x_t)$ into the hidden state h_t to predict the next item x_{t+1}. The input and output share the same embedding, so $e_i = E[i]$ represents item i both in the history and as a candidate during prediction. We use sampled softmax to avoid normalizing over the entire item catalog. For each prediction position $t \in S$, let $\mathcal{N}_t$ denote the sampled negatives. The temperature-scaled cosine logit and target probability are

$$z_t(i) = \frac{1}{\tau} \left\langle \frac{h_t}{\|h_t\|}, \frac{e_i}{\|e_i\|} \right\rangle, \qquad p_t(x_{t+1}) = \frac{\exp z_t(x_{t+1})}{\exp z_t(x_{t+1}) + \sum_{n\in\mathcal{N}_t} \exp z_t(n)}.$$

We minimize the average negative log-likelihood over all positions in $\mathcal{D}$:

$$\mathcal{L}_{\text{NTP}} = -\frac{1}{\sum_{S\in\mathcal{D}} |S|} \sum_{S\in\mathcal{D}} \sum_{t\in S} \log p_t(x_{t+1}). \tag{1}$$

Transfer and freezing. After pretraining, we transfer and freeze only the item embedding matrix E and reinitialize all remaining parameters ϕ. The downstream model is then trained with the same CTR objective:

$$\min_{\phi} \mathcal{L}_{\text{disc}}(E, \phi). \tag{2}$$

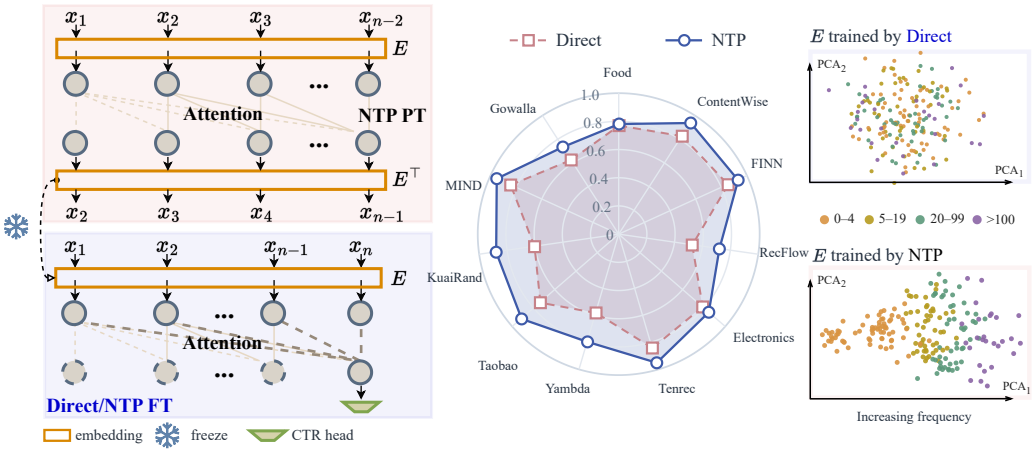


Figure 1: Training routes of Direct and NTP methods. **Left:** NTP pretraining learns the item embeddings, which are then transferred and frozen for downstream CTR training; Direct trains its item embeddings and CTR reader from random initialization. **Middle:** Mean AUC for Direct and NTP on all eleven main-table datasets. **Right:** PCA projections of embeddings for KuaiRand items trained by Direct and NTP, colored by item frequency.

We refer to this configuration as NTP. Figure 1 compares the two training routes and shows that NTP outperforms direct training on most displayed datasets. Since only the item embeddings are

transferred, these gains indicate that the pretrained embeddings provide useful information to a newly initialized downstream model. We next examine their structure and the gradient paths that shape them.

3.2 Gradient Paths in the Transferred Item Table

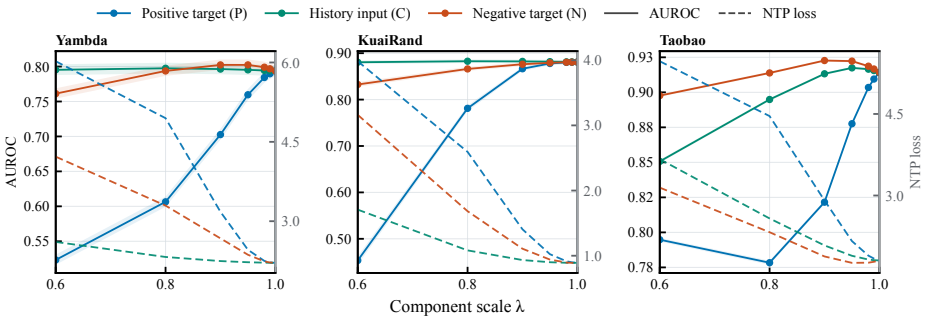


Figure 2: Retained-component scaling on Yambda, KuaiRand, and Taobao. Holding the downstream model fixed, each color scales one retained NTP update component: positive target (P), history input (C), or negative target (N). Solid curves and the left axes report downstream AUROC; shading gives its pointwise 95% interval conditional. Dashed curves and the right axes report sampled-softmax NTP loss.

During NTP training, an item can serve as the positive next-item target, a sampled negative candidate, or a historical input used to predict subsequent items. Because inputs and outputs share the same embedding matrix, gradients from these three computational paths accumulate in the corresponding item embedding. By the chain rule, the gradient of item embedding e_i can be decomposed as

$$\nabla_{e_i}\mathcal{L}_{\mathrm{NTP}} = \underbrace{\nabla^{P}_{e_i}\mathcal{L}_{\mathrm{NTP}}}_{\text{positive-target output}} + \underbrace{\nabla^{N}_{e_i}\mathcal{L}_{\mathrm{NTP}}}_{\text{negative-target output}} + \underbrace{\nabla^{C}_{e_i}\mathcal{L}_{\mathrm{NTP}}}_{\text{historical input}}. \tag{3}$$

The positive-target gradient acts through the true next item's output score and encourages the model to increase its predicted probability. The negative-target gradient acts through sampled candidate scores and encourages the model to reduce them relative to the positive target. The historical-input gradient propagates through the history encoder, adjusting input embeddings to support subsequent predictions. For example, given a sequence (a, b, c), the model predicts c from (a, b) and samples d as a negative candidate. Then e_c receives positive-target gradient, e_d receives N, and e_a and e_b receive historical-input gradient. An item can assume different roles across prediction tasks or serve as both a historical input and an output candidate within the same task.

To assess sensitivity to different retained update components, we record the retained positive-target, historical-input, and negative-target update components during NTP pretraining. Each component tracks the contribution from one path that remains in the final item embeddings along the observed training trajectory (Appendix C). We then keep the downstream model's other parameters fixed and scale one retained component at a time by λ, using the modified embeddings for both history and candidate lookups. Here, $\lambda = 1$ recovers the original pretrained embeddings, while lower values attenuate the selected component.

Figure 2 shows that reducing the positive-target component tends to cause the largest decline in downstream AUC. It also produces the largest and most consistent increase in the matched NTP-loss diagnostic. In contrast, reducing the historical-input or negative-target component usually has a smaller effect, and modest reductions can slightly improve AUC near $\lambda = 1$. These component-scaling interventions show that the downstream model relies strongly on the retained positive-target component for prediction. This dependence motivates examining how frequently each item receives positive-target supervision and whether strengthening it for infrequent targets can improve transfer.

4 FREQUENCY ADJUSTMENT

4.1 METHOD

Section 3 decomposed the gradient to the shared item table into positive-target, negative-target, and history-input paths. Based on this decomposition and the component-scaling analysis, we propose Frequency Adjustment (FA), which scales only the positive-target path in NTP pretraining. Let f_i be the number of times item i appears as an observed next-item target. FA assigns each item a multiplier $r_i = r(f_i)$, larger for less frequent targets, and scales its positive-target gradient by r_i. Its history-input and negative-target paths, forward scores, and NTP loss remain unchanged.

At the same forward state, the adjusted gradient to e_i is

$$\widetilde{\nabla}_{e_i}\mathcal{L}_{\mathrm{NTP}} = r_i\nabla^P_{e_i}\mathcal{L}_{\mathrm{NTP}} + \nabla^C_{e_i}\mathcal{L}_{\mathrm{NTP}} + \nabla^N_{e_i}\mathcal{L}_{\mathrm{NTP}}. \tag{4}$$

The three terms on the right are the original path gradients at that forward state. FA does not weight the entire prediction loss or change the gradient sent to the predictive state. Later training states can differ as the parameter updates accumulate. Specifically, reweighting the loss and adjusting pretrained embeddings underperformed FA in preliminary tests and were not adopted (Appendix D.4).

4.2 TRANSFER BENEFITS VARY WITH RELATIVE CANDIDATE FREQUENCY

An item receives a gradient through the positive-target path only when it appears as the observed next-item target. Its target frequency therefore measures its opportunities to receive positive-target supervision during pretraining. We examine how NTP's gains over Direct and FA's additional gains over NTP vary with the target frequencies of positive and negative candidates.

We characterize an item's frequency by the number of times it appears as a positive target in the pretraining data. We compare Direct, NTP, and FA on the same candidate pairs. Each pair consists of a positive example (u_q, H_q, i_q^+) and a negative example (u_q, H_q, i_q^-) sharing the same user history. For each pair, we compute the score margin and the corresponding pairwise loss:

$$m_M(q) = s_M(u_q, H_q, i_q^+) - s_M(u_q, H_q, i_q^-), \qquad \ell_M(q) = \log(1 + \exp(-m_M(q)/T_M)). \tag{5}$$

Here, M denotes one of three methods: Direct, NTP, or FA; s_M is the corresponding downstream logit. For each method and seed, we fit T_M on a disjoint calibration set and hold it fixed across frequency groups to remove the effect of method-specific logit scales from the comparison. We then bin positive and negative candidates by their target frequencies. Let Q_{ab} denote the set of candidate pairs whose positive and negative frequencies fall in bins a and b, respectively. We define the mean loss within each bin pair and the relative loss reduction of NTP over Direct as

$$L_M(a,b) = \frac{1}{|Q_{ab}|}\sum_{q\in Q_{ab}} \ell_M(q), \qquad G_{ab} = 1 - \frac{L_{\mathrm{NTP}}(a,b)}{L_{\mathrm{Direct}}(a,b)}. \tag{6}$$

Figure 3(a) compares NTP's gains over Direct across datasets. It indicates that NTP yields much smaller improvements when distinguishing an infrequent positive from a frequent negative than when distinguishing a frequent positive from an infrequent negative. Thus, NTP's overall transfer advantage is uneven across candidate-frequency combinations.

Building on the component-scaling analysis in Section 3.2, we further examine how the retained positive-target component varies with target frequency. For each of the three datasets, Figure 3(b) groups items by target frequency and reports, under NTP, the normalized signed projection of this component onto the total embedding displacement within each group. Under NTP, this measure is smallest in the lowest-frequency group on all three datasets. This indicates that frequency disparities among items lead to inconsistent learning of this component. This observation motivates examining whether enhancing positive-target updates for low-frequency items can improve transfer performance.

In Figure 3(a), FA's incremental pairwise-loss gains are particularly pronounced in rare-positive/frequent-negative comparisons on KuaiRand and Taobao, while gains on Yambda are more broadly distributed. This indicates that the FA method alleviates the frequency degradation issue associated with direct NTP training and yields overall performance improvements on certain datasets.

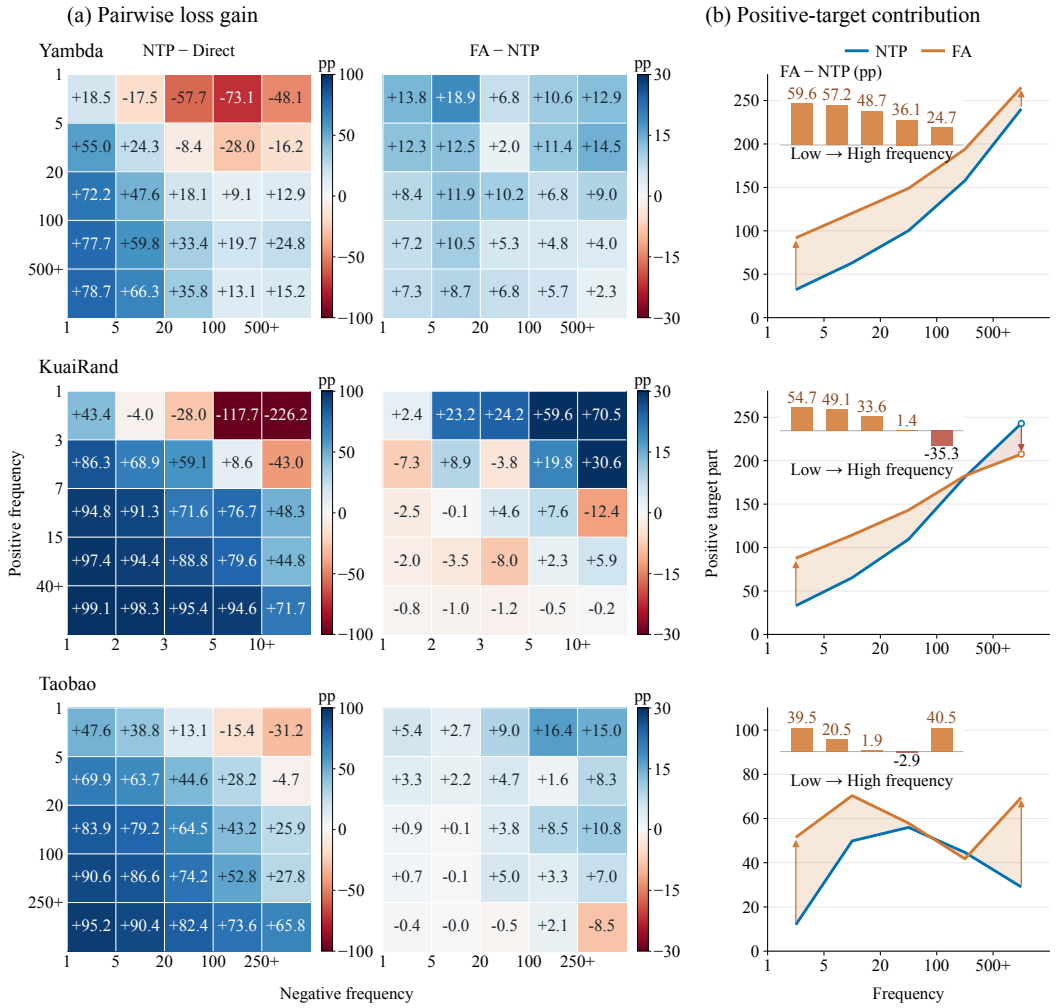


Figure 3: Frequency-dependent transfer gains and positive-target components on Yambda, KuaiRand, and Taobao. **(a)** Calibrated pairwise-loss gains of NTP over Direct and the incremental gains of FA over NTP, both normalized by the same Direct loss. Rows and columns group positive and negative candidates, respectively, by pretraining target frequency. **(b)** Normalized signed projections of the retained positive-target component from pretraining trajectories on different datasets. Shading and bars show the observed FA−NTP differences.

To examine whether FA balances the positive target components of different frequencies as expected, we plotted the impact of the retained forward target projection after applying FA, as shown in Figure 3(b). Across all three datasets, FA increased this projection for the lowest-frequency group. For the Yambda and KuaiRand datasets, the magnitude of the increase was greater in the low-frequency group than in the high-frequency group, whereas the Taobao dataset exhibited a non-monotonic pattern of change. Overall, however, the FA method narrowed the gap between low- and high-frequency groups, thereby mitigating frequency-induced inconsistencies.

5 Shared-Query Cross Attention

FA adjusts the strength of positive-target gradients for infrequent items. Our second method, Cross, changes the predictive state h through which the observed history shapes their direction. For a history prefix $S_{\le t} = (x_1, \ldots, x_t)$ and next-item target $y = x_{t+1}$, let $\ell = -\log p(y)$, $\widehat{h} = h/\|h\|_2$, and

$\widehat{e}_y = e_y/\|e_y\|_2$. With the cosine scoring head, the positive-target output path gives

$$\nabla^P_{e_y} \ell = \frac{p(y) - 1}{\tau \|e_y\|_2} \left(I - \widehat{e}_y \widehat{e}_y^\top\right) \widehat{h}. \tag{7}$$

For a fixed target embedding, the descent direction follows the projection of $\widehat{h}$ orthogonal to $\widehat{e}_y$. How the history is aggregated into h therefore determines the direction of the positive-target gradient. In standard NTP, the latest item's embedding initializes the attention query and enters the residual stream directly, giving that item a direct influence on h.

Cross replaces this item-specific query and residual initialization with a learned vector b shared across users and positions, as in XLNet's query stream (Yang et al., 2019). Unlike XLNet's two-stream architecture, Cross uses cross-attention only in the first layer, where b attends to the observed item embeddings; subsequent layers use self-attention over the resulting states. Let $\mathbf{e} = E[S_{\le t}] = (e_{x_1}, \ldots, e_{x_t})$ be the input embedding sequence, $\mathbf{b} = (b, \ldots, b)$ the shared initial queries, and $\mathbf{h}^{(\ell)}$ the state sequence after layer ℓ. The K-layer encoder is

$$\mathbf{h}^{(1)} = \mathrm{CrossAttn}(\mathbf{b}, \mathbf{e}), \mathbf{h}^{(\ell)} \quad = \mathrm{SelfAttn}_\ell\left(\mathbf{h}^{(\ell-1)}\right), \qquad \ell = 2, \ldots, K. \tag{8}$$

Both blocks use causal attention with standard feed-forward and residual operations. The final state at the last visible position, after normalization, gives h. The latest item remains available through attention but no longer initializes the query or residual stream. Cross retains chronological next-item prediction rather than permutation language modeling to learn item embeddings for frozen-table CTR transfer.

To assess the predictive state's sensitivity to individual history items, we perturb one input embedding at a time in NTP and Cross, with model parameters fixed. The perturbation $\boldsymbol{\delta}_j$ has norm α times the median current-item embedding norm at NTP initialization. Let h and $h^{(j)}_\alpha$ be the predictive states for $\mathbf{e}$ and $\mathbf{e}^{(j)}_\alpha = (e_{x_1}, \ldots, e_{x_j} + \boldsymbol{\delta}_j, \ldots, e_{x_t})$, respectively. We measure

$$D_j(\alpha) = 1 - \cos\left(h, h^{(j)}_\alpha\right). \tag{9}$$

We compare the last 32 visible positions on KuaiRand at $\alpha = 0.1$ (Figure 4(a)), and the last two positions across perturbation strengths on KuaiRand, Yambda, and Taobao (Figure 4(b1)–(b3)). Each trial perturbs one position in the original full history.

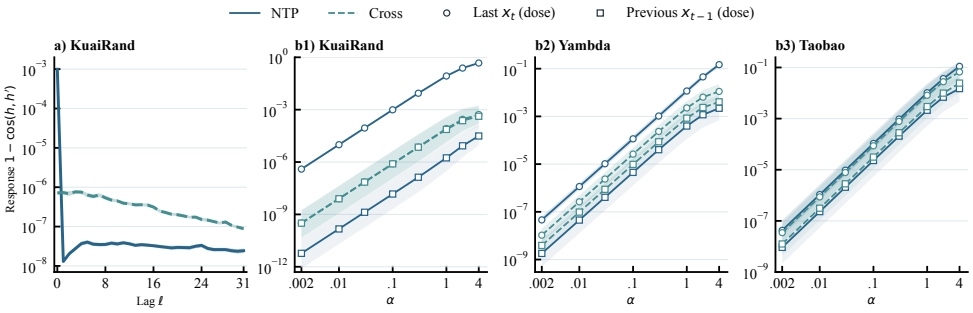


Figure 4: Changes in the predictive state induced by perturbing a single input position. **(a)** Responses at individual positions on KuaiRand at $\alpha = 0.1$; lag $\ell = 0$ denotes the last visible item. **(b1)–(b3)** Dose-response curves for KuaiRand, Yambda, and Taobao. Circles indicate the last position x_t, and squares indicate the preceding position x_{t-1}; blue solid lines denote NTP, and cyan dashed lines denote Cross.

Figure 4 examines the predictive state's response to individual history items across positions and perturbation strengths. Figure 4(a) fixes $\alpha = 0.1$ on KuaiRand: NTP's response to perturbing the last visible item is much larger than its responses at the preceding 31 positions, whereas Cross has a lower response that varies less across positions. Figure 4(b1)–(b3) further compares the dose-response

curves for the last position and its immediate predecessor. NTP is more sensitive to perturbations at the last position on all three datasets; Cross markedly reduces the last-position response on KuaiRand and Yambda, while the difference between the two models is relatively small on Taobao. Together, the experiments across positions and perturbation strengths show that Cross alleviates excessive sensitivity to the last position.

6 EXPERIMENTS

Table 1: AUC on eleven datasets. Mean Best AUC is shown above one-significant-digit sample SD. The small Joint–NTP Δ appears at the lower right of the Joint mean; its mean weights datasets equally. Bold and underline mark the highest and second-highest displayed means in each dataset, respectively.

Dataset	Direct	DIN	DeepFM	CAN	NTP	Cross	FA	Joint
Electronics	0.7848	0.7872	0.7970	0.7946	0.8421	0.8420	<u>0.8639</u>	**0.8649**$_{(\Delta+0.023)}$
	±0.002	±0.002	±0.0009	±0.0001	±0.002	±0.003	±0.002	±0.004
Food	0.7726	0.7201	0.7223	0.7251	0.7839	0.7871	**0.7955**	<u>0.7942</u>$_{(\Delta+0.010)}$
	±0.002	±0.0008	±0.0002	±0.00002	±0.0008	±0.0009	±0.001	±0.0006
ContentWise	0.8292	0.8299	0.8357	0.8390	0.9402	<u>0.9623</u>	0.9439	**0.9718**$_{(\Delta+0.032)}$
	±0.003	±0.005	±0.001	±0.001	±0.002	±0.001	±0.002	±0.0009
FINN	0.8506	0.8412	0.8332	0.8440	0.9286	0.9299	**0.9344**	<u>0.9342</u>$_{(\Delta+0.006)}$
	±0.005	±0.0007	±0.0002	±0.002	±0.0002	±0.0004	±0.002	±0.0009
RecFlow	0.5244	0.5215	0.5134	0.5008	0.7188	<u>0.7493</u>	0.7379	**0.7949**$_{(\Delta+0.076)}$
	±0.003	±0.001	±0.01	±0.0005	±0.007	±0.005	±0.006	±0.002
Tenrec	0.8391	0.8362	0.8497	0.8617	0.9483	0.9509	<u>0.9590</u>	**0.9606**$_{(\Delta+0.012)}$
	±0.004	±0.005	±0.006	±0.003	±0.001	±0.001	±0.001	±0.0005
Yambda	0.5799	0.6345	0.5918	0.5803	0.7938	0.8047	<u>0.8168</u>	**0.8173**$_{(\Delta+0.023)}$
	±0.003	±0.00003	±0.005	±0.005	±0.002	±0.003	±0.003	±0.002
Taobao	0.7387	0.7730	0.7171	0.7287	0.9145	0.9142	**0.9229**	<u>0.9213</u>$_{(\Delta+0.007)}$
	±0.004	±0.0006	±0.0007	±0.001	±0.0006	±0.0004	±0.0008	±0.0003
KuaiRand	0.6057	0.8429	0.7811	0.5122	0.8809	0.8886	<u>0.8894</u>	**0.8956**$_{(\Delta+0.015)}$
	±0.02	±0.003	±0.009	±0.01	±0.0005	±0.001	±0.001	±0.0005
MIND	0.8453	0.8515	0.8916	0.8526	0.9540	0.9563	<u>0.9592</u>	**0.9609**$_{(\Delta+0.007)}$
	±0.009	±0.006	±0.004	±0.004	±0.004	±0.005	±0.003	±0.004
Gowalla	0.6287	0.5778	0.6193	0.5806	0.7378	0.7437	<u>0.7457</u>	**0.7679**$_{(\Delta+0.030)}$
	±0.005	±0.02	±0.004	±0.002	±0.004	±0.003	±0.007	±0.002
							Mean Δ (11 datasets)	+0.022

6.1 EXPERIMENTAL SETUP

For Direct, NTP, Cross, FA, and their combination (Cross+FA), we use a single-layer, single-head Transformer with a hidden dimension of 128 and an effective batch size of 512 in each applicable training stage. Each training stage runs for 10 epochs with a maximum history length of 512. FA and Cross+FA use the fixed multiplier profile of Section 4.1, specified a priori and shared across datasets. We report the mean and standard deviation over five random seeds.

Following Wang et al. (2025), we use Best AUC with validation-based checkpoint selection: each dataset is split into training, validation, and test sets in an 8:1:1 ratio, and for each run, we select the downstream checkpoint with the highest validation AUC and report its AUC on the held-out test set since different sets of pretrained item embeddings may reach their best validation performance at different stages of downstream training, while continued optimization can lead to overfitting and a subsequent decline in AUC.

To broaden the architectural comparison, we complement the Transformer-based Direct baseline with three standard CTR models: DeepFM (Guo et al., 2017), DIN (Zhou et al., 2018), and CAN (Bian et al., 2022). Their training configurations are based on the respective original papers and reference implementations, with adaptations to our input setting.

6.2 Main Results

Table 1 reports Direct, DIN, DeepFM, CAN, NTP, Cross, FA, and Cross+FA on the evaluated datasets. Learned methods are compared by their mean test AUC across seeds at validation-selected downstream checkpoints. The datasets are documented in Appendix A.

First, directly trained CTR baselines—such as DeepFM, DIN, and CAN—significantly underperform compared to pre-training and fine-tuning approaches (exemplified by NTP) across most datasets. This indicates that the limited downstream supervision signals inherent in discriminative tasks prevent the model from learning adequately. In contrast, NTP unlocks the model's potential and achieves superior downstream performance, highlighting the crucial role of embedding structure in downstream tasks.

Second, FA and Cross provide complementary gains over standard NTP: FA improves mean AUC on every dataset, while Cross improves it on most. Their combination, Cross+FA, surpasses NTP on every dataset and ranks first or second among all compared methods, generally outperforming either modification alone. These results support the combined value of adjusting positive-target supervision and the construction of the predictive state during pretraining. The gains carry over to downstream CTR prediction through frozen item embeddings, demonstrating that the proposed changes improve the usefulness of the pretrained representations.

Moreover, Table 2 shows that the benefits extend to deeper pretraining encoders. Cross and FA both improve mean downstream AUC over NTP at the same depth. Cross+FA achieves the largest average gain at every evaluated depth, supporting the effectiveness and complementarity of the two modifications in deeper models.

Table 2: Average difference in downstream AUC relative to NTP at the same pretraining depth, with checkpoints selected by validation AUC. Results are averaged over datasets used in Table 1.

Pretraining method	1L	2L	4L
Cross	$+0.0078$	$+0.0080$	$+0.0069$
FA	$+0.0114$	$+0.0111$	$+0.0138$
Joint	$+0.0219$	$+0.0167$	$+0.0153$

7 Conclusion and Limitations

We investigated how NTP pretraining learns item representations that transfer to CTR prediction, with a focus on positive-target supervision and the predictive state through which it operates. Our analysis highlights item frequency and hidden-state construction as two important factors shaping this supervision. These findings motivate two complementary modifications: FA strengthens the positive-target learning signal for infrequent items, while Cross uses a learned shared query to aggregate the causal history, reducing the predictive state's excessive sensitivity to the most recent interaction. Experiments on public datasets show that both methods improve downstream CTR performance, yield complementary gains when combined, and retain average AUC advantages with deeper pretraining encoders. These results highlight the value of refining how items receive supervision and how historical context informs prediction when learning transferable item representations.

Our evaluation is limited to offline CTR prediction on the studied datasets with frozen item embeddings. The effectiveness of FA and Cross in online recommendation therefore remains to be validated. Further evaluation in deployed systems and under alternative transfer settings, such as end-to-end fine-tuning, would help establish the broader applicability of these findings.

AI USE STATEMENT

In this work, we used generative AI tools solely to assist with translation of the manuscript. We did not use generative AI tools for any other tasks for which disclosure is required or recommended. We have reviewed all translations to ensure that they accurately preserve the intended meaning and technical content of the original text. We take full responsibility for the final content of this work, including all text translated with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed task descriptions, explanations of the theoretical analysis, and experimental settings in the main paper and the appendix. Readers can refer to these materials for the task formulations, theoretical reasoning, and experimental procedures underlying our findings. Together, these descriptions provide sufficient information to reproduce the experimental phenomena reported in the main paper.

REFERENCES

Himan Abdollahpouri, Robin Burke, and Bamshad Mobasher. Controlling popularity bias in learning-to-rank recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pp. 42–46. Association for Computing Machinery, 2017. doi: 10.1145/3109859.3109912.

Alibaba Group. User behavior data from taobao for recommendation. Alibaba Tianchi Open Dataset, 2018. URL https://tianchi.aliyun.com/dataset/649. Accessed: 2026-09-03.

Weijie Bian, Kailun Wu, Lejian Ren, Qi Pi, Yujing Zhang, Can Xiao, Xiang-Rong Sheng, Yong-Nan Zhu, Zhangming Chan, Na Mou, Xinchen Luo, Shiming Xiang, Guorui Zhou, Xiaoqiang Zhu, and Hongbo Deng. CAN: Feature co-action network for click-through rate prediction. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*, pp. 57–65, 2022. doi: 10.1145/3488560.3498435.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Renqin Cai, Dawei Sun, Yuanjun Yao, Zhiyong Wang, Velvin Fu, Maggie Zhuang, Yu Shi, Zhongnan Fang, Xuan Cao, Jing Qian, and Rui Li. WHALE: A scalable unified model for recommendation with Wukong-HSTU architecture. arXiv preprint arXiv:2607.17017, 2026. URL https://arxiv.org/abs/2607.17017.

Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1082–1090. Association for Computing Machinery, 2011. doi: 10.1145/2020408.2020579. URL https://snap.stanford.edu/data/loc-gowalla.html.

Simen Eide, Arnoldo Frigessi, Helge Jenssen, David S. Leslie, Joakim Rishaug, and Sofie Verrewaere. FINN.no slates dataset: A new sequential dataset logging interactions, all viewed items and click responses/no-click for recommender systems research. In *Fifteenth ACM Conference on Recommender Systems*, pp. 556–558. Association for Computing Machinery, 2021. doi: 10.1145/3460231.3474607.

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 and Knowledge*

Management, pp. 3953–3957. Association for Computing Machinery, 2022. doi: 10.1145/3511808.3557624.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. In *International Conference on Learning Representations*, 2019.

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*, 2022. doi: 10.1145/3523227.3546767. URL https://arxiv.org/abs/2203.13366.

Chengyue Gong, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Frage: Frequency-agnostic word representation. In *Advances in Neural Information Processing Systems*, 2018.

Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 1725–1731, 2017. doi: 10.24963/ijcai.2017/239.

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. doi: 10.1145/3534678.3539381.

Yuchen Jiang, Jie Zhu, Xintian Han, Hui Lu, Kunmin Bai, Mingyu Yang, Shikang Wu, Ruihao Zhang, Wenlin Zhao, Shipeng Bai, Sijin Zhou, Huizhi Yang, Tianyi Liu, Wenda Liu, Ziyan Gong, Haoran Ding, Zheng Chai, Deping Xie, Zhe Chen, Yuchao Zheng, and Peng Xu. TokenMixer-Large: Scaling up large ranking models in industrial recommenders. arXiv preprint arXiv:2602.06563, 2026. URL https://arxiv.org/abs/2602.06563.

Keito Kozaki, Keigo Sakurai, Ren Togo, Takahiro Ogawa, and Miki Haseyama. Residual dominance as a structural account of last-item reliance in causal self-attention recommenders. In *Proceedings of the 20th ACM Conference on Recommender Systems*, 2026. doi: 10.1145/3773078.3831798. URL https://arxiv.org/abs/2608.14021. Accepted at RecSys 2026.

Xiaodong Li, Alin Fan, Mingyang Li, Yan Xiao, Shichao Nie, Junfeng Zhang, Shaochuan Lin, Zhanming Ou, Tao Luo, and Xiaoyi Zeng. LazFormer: Scaling transformers for industrial recommendation via transferable generative pre-training, 2026. URL https://arxiv.org/abs/2609.14978.

Jianghao Lin, Yanru Qu, Wei Guo, Xinyi Dai, Ruiming Tang, Yong Yu, and Weinan Zhang. MAP: A model-agnostic pretraining framework for click-through rate prediction. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023. doi: 10.1145/3580305.3599422. URL https://arxiv.org/abs/2308.01737.

Qi Liu, Kai Zheng, Rui Huang, Wuchao Li, Kuo Cai, Yuan Chai, Yanan Niu, Yiqun Hui, Bing Han, Na Mou, Hongning Wang, Wentian Bao, Yun Yu, Guorui Zhou, Han Li, Yang Song, Defu Lian, and Kun Gai. RecFlow: An industrial full flow recommendation dataset. In *International Conference on Learning Representations*, pp. 40368–40385, 2025.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 43–52. Association for Computing Machinery, 2015. doi: 10.1145/2766462.2767755.

Jiaqi Mu and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*, 2018.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pp. 188–197. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1018. URL https://aclanthology.org/D19-1018/.

Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pp. 11–18. Association for Computing Machinery, 2008. doi: 10.1145/1454008.1454012.

Fernando B. Pérez Maurera, Maurizio Ferrari Dacrema, Lorenzo Saule, Mario Scriminaci, and Paolo Cremonesi. ContentWise Impressions: An industrial dataset with impressions included. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 3093–3100. Association for Computing Machinery, 2020. doi: 10.1145/3340531.3412774.

Alexander Ploshkin, Vladislav Tytskiy, Alexey Pismenny, Vladimir Baikalov, Evgeny Taychinov, Artem Permiakov, Daniil Burlakov, and Eugene Krofto. Yambda-5B: A large-scale multi-modal dataset for ranking and retrieval. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pp. 894–901. Association for Computing Machinery, 2025. doi: 10.1145/3705328.3748163.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2022. doi: 10.1145/3534678.3539253. URL https://arxiv.org/abs/2206.12811.

Chunqi Wang, Bingchao Wu, Zheng Chen, Lei Shen, Bing Wang, and Xiaoyi Zeng. Scaling transformers for discriminative recommendation via generative pretraining. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, 2025. doi: 10.1145/3711896.3737117. URL https://arxiv.org/abs/2506.03699.

Chunqi Wang, Bingchao Wu, Taotian Pang, Jiahao Wang, Jie Yang, Jia Liu, Hao Zhang, Hai Zhu, Lei Shen, Shizhun Wang, Bing Wang, and Xiaoyi Zeng. SORT: A systematically optimized ranking transformer for industrial-scale recommenders. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management*, 2026. doi: 10.1145/3799682.3840109. URL https://arxiv.org/abs/2603.03988.

Ruoxi Wang, Rakesh Shivanna, Derek Cheng, Sagar Jain, Dong Lin, Lichan Hong, and Ed Chi. DCN V2: Improved deep & cross network and practical lessons for web-scale learning to rank systems. In *Proceedings of the Web Conference 2021*, pp. 1785–1797. Association for Computing Machinery, 2021. doi: 10.1145/3442381.3450078. URL https://arxiv.org/abs/2008.13535.

Chuhan Wu, Fangzhao Wu, Tao Qi, Jianxun Lian, Yongfeng Huang, and Xing Xie. PTUM: Pre-training user model from unlabeled user behaviors via self-supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1939–1944, 2020a. doi: 10.18653/v1/2020.findings-emnlp.174. URL https://aclanthology.org/2020.findings-emnlp.174/.

Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, and Ming Zhou. MIND: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 3597–3606. Association for Computational Linguistics, 2020b. doi: 10.18653/v1/2020.acl-main.331. URL https://aclanthology.org/2020.acl-main.331/.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/hash/dc6a7e655d7e5840e66733e9ee67cc69-Abstract.html.

Junjie Yao, Zhongwang Zhang, and Zhi-Qin John Xu. An analysis for reasoning bias of language models with small initialization, 2025. URL https://arxiv.org/abs/2502.04375.

Junjie Yao, Liangkai Hang, and Zhi-Qin John Xu. Context staircase: Signature-aligned dynamics of token embeddings under small initialization, 2026. URL https://arxiv.org/abs/2608.30315.

Sangwon Yu, Jongyoon Song, Heeseung Kim, Seongmin Lee, Woo-Jong Ryu, and Sungroh Yoon. Rare tokens degenerate all tokens: Improving neural text generation via adaptive gradient gating for rare token embeddings. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 29–45, 2022. doi: 10.18653/v1/2022.acl-long.3. URL https://aclanthology.org/2022.acl-long.3/.

Guanghu Yuan, Fajie Yuan, Yudong Li, Beibei Kong, Shujie Li, Lei Chen, Min Yang, Chenyun Yu, Bo Hu, Zang Li, Yu Xu, and Xiaohu Qie. Tenrec: A large-scale multipurpose benchmark dataset for recommender systems. In *Advances in Neural Information Processing Systems*, volume 35, pp. 11480–11493. Neural Information Processing Systems Foundation, 2022. doi: 10.52202/068431-0834.

Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Shen Li, Yanli Zhao, Yuchen Hao, Yantao Yao, Ellie Dingqiao Wen, Jongsoo Park, Maxim Naumov, and Wenlin Chen. Wukong: Towards a scaling law for large-scale recommendation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 59421–59434. PMLR, 2024. URL https://proceedings.mlr.press/v235/zhang24ao.html.

Zhaoqi Zhang, Haolei Pei, Jun Guo, Tianyu Wang, Yufei Feng, Hui Sun, Shaowei Liu, and Aixin Sun. OneTrans: Unified feature interaction and sequence modeling with one transformer in industrial recommender. In *Proceedings of the ACM Web Conference 2026*. Association for Computing Machinery, 2026. URL https://arxiv.org/abs/2510.26104.

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*, KDD '18, pp. 1059–1068. ACM, 2018. doi: 10.1145/3219819.3219823. URL https://doi.org/10.1145/3219819.3219823.

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. doi: 10.1609/aaai.v33i01.33015941.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S^3-Rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pp. 1893–1902, 2020. doi: 10.1145/3340531.3411954.

Jie Zhu, Zhifang Fan, Xiaoxie Zhu, Yuchen Jiang, Hangyu Wang, Xintian Han, Haoran Ding, Xinmin Wang, Wenlin Zhao, Zhen Gong, Huizhi Yang, Zheng Chai, Zhe Chen, Yuchao Zheng, Qiwei Chen, Feng Zhang, Xun Zhou, Peng Xu, Xiao Yang, Di Wu, and Zuotao Liu. RankMixer: Scaling up ranking models in industrial recommenders. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, 2025. doi: 10.1145/3746252.3761507. URL https://arxiv.org/abs/2507.15551.

A Evaluation Protocol and Training-only Popularity Baseline

Datasets and sources. The eleven datasets in Table 1 are Amazon Electronics and Amazon Grocery and Gourmet Food (Food) (McAuley et al., 2015; Ni et al., 2019); ContentWise Impressions (ContentWise) (Pérez Maurera et al., 2020); FINN.no Slates (FINN) (Eide et al., 2021); RecFlow (Liu et al., 2025); Tenrec (the QB-video subset) (Yuan et al., 2022); Yambda (the 50M subset) (Ploshkin et al., 2025); Taobao UserBehavior (Alibaba Group, 2018); KuaiRand (the 27K release) (Gao et al., 2022); MIND (the Small subset) (Wu et al., 2020b); and Gowalla check-ins (Cho et al., 2011). These references identify the original data sources; the evaluation protocol below and Table 3 describe the splits, labels, and negative candidates used in our experiments.

Main comparison evaluation protocol. Following Section 6, each dataset uses an 8:1:1 training/validation/test split shared by the compared methods. Training histories supply pretraining inputs, and labeled training examples supply the downstream binary loss. For each run, the downstream checkpoint with the highest validation ROC AUC is selected, and its AUC on the held-out test partition is reported as Best AUC. Test AUC is not used for checkpoint selection. The main comparisons report the mean and sample standard deviation over five random seeds. Appendix E specifies the common training settings, candidate sampling, and transfer procedure; fixed-seed and fixed-endpoint diagnostics state their own scopes.

Table 3: Positive-label sources and downstream negative candidates in the eleven-dataset panel. Sampled negatives are constructed labels, not observed non-clicks. Dataset preprocessing applies its own eligibility and exclusion rules.

Dataset	Positive source	Negative source
Electronics, Food	Product reviews treated as interactions	Uniform training-catalog sample
Taobao, Gowalla	Behavior events; check-ins	Uniform training-catalog sample
ContentWise	Implicit content interactions	Uniform training-catalog sample
FINN	Clicked slate items	Uniform training-catalog sample
RecFlow	Logged clicks	Uniform training-catalog sample
Tenrec QB-video	Published click labels	Uniform training-catalog sample
KuaiRand-27K	Clicked videos	Uniform clicked-video-catalog sample
MIND Small	Clicked news items	Uniform training-catalog sample
Yambda-50M	Positive listening or feedback	Eligible explicit dislike, otherwise uniform training-catalog sample

These downstream candidates are serialized before model training and reused by the compared methods. For example, Taobao excludes items in the current sequence window from sampled negatives, while the KuaiRand construction excludes the current positive target but can sample an item already present in the user's history. This dataset dependence matters when interpreting frequency-conditioned AUC and pairwise loss. Pretraining's 4,096 sequence-shared sampled negatives are a different candidate mechanism, specified in Appendix E.

Training-only popularity baseline. Pop assigns a candidate its occurrence count in training histories with the final history item removed. Let $\mathcal{T}^+$ index positive downstream-training rows and H_r be a row's serialized history, excluding the downstream target. Define P_r by removing the final item of H_r, filtering out-of-vocabulary items, and retaining at most the last $L = 512$ items. Then

$$\mathrm{Pop}(i) = \sum_{r \in \mathcal{T}^+} \sum_{x \in P_r} \mathbf{1}[x = i].$$

Downstream targets and paired negative rows never contribute to the counts. Each source history is counted once; repeated occurrences and overlapping windows retain their training-exposure multiplicity. In particular, Yambda has 33,693 training histories from 6,920 users. Validation and test histories, targets, and labels do not affect fitting, and unseen items receive zero.

Fixed scores are evaluated on all saved downstream test rows and their original negative candidates, using the loader's vocabulary filtering and minimum history length of two. We report global ROC AUC for the fixed rule that ranks larger training counts higher, giving half credit to tied positive–negative scores. The score direction is fixed before evaluation, and AUC values below 0.5 are

Table 4: Training-only Pop test AUC and test support for the eleven main-panel datasets. Counts use training histories with their final item removed; downstream targets are excluded. ROC AUC is computed with larger counts ranked higher.

Dataset	Pop AUC	Training occurrences	Test rows
Electronics	0.586287	785,571	115,368
Food	0.629098	568,060	76,320
ContentWise	0.868455	1,837,988	25,138
FINN	0.806075	3,472,132	417,142
RecFlow	0.394132	2,840,099	19,800
Tenrec QB-video	0.883051	792,106	19,242
Yambda-50M	0.651986	15,205,931	60,758
Taobao	0.785931	9,363,724	396,084
KuaiRand-27K	0.853558	9,620,366	16,370
MIND Small	0.502880	1,368,764	53,924
Gowalla	0.462871	2,760,286	53,756

retained. Exact integer pair-count AUC agrees with an independent rank-based computation. Pop has no learned parameters or training-seed variability, so no seed SD is reported.

B Supplement to Section 3

B.1 Item-Embedding PCA on Three Datasets

For Yambda, KuaiRand, and Taobao, we apply PCA to the row-normalized item embeddings of Direct, NTP, and Joint (Cross+FA), fitting each projection independently on the full training-candidate pool. Within each dataset, all three panels show the same 1,966 items, colored by their pretraining target-frequency groups. The axes have independent bases and arbitrary signs. PCA is used only for visualization; neither the projected coordinates nor the frequency colors enter downstream CTR training.

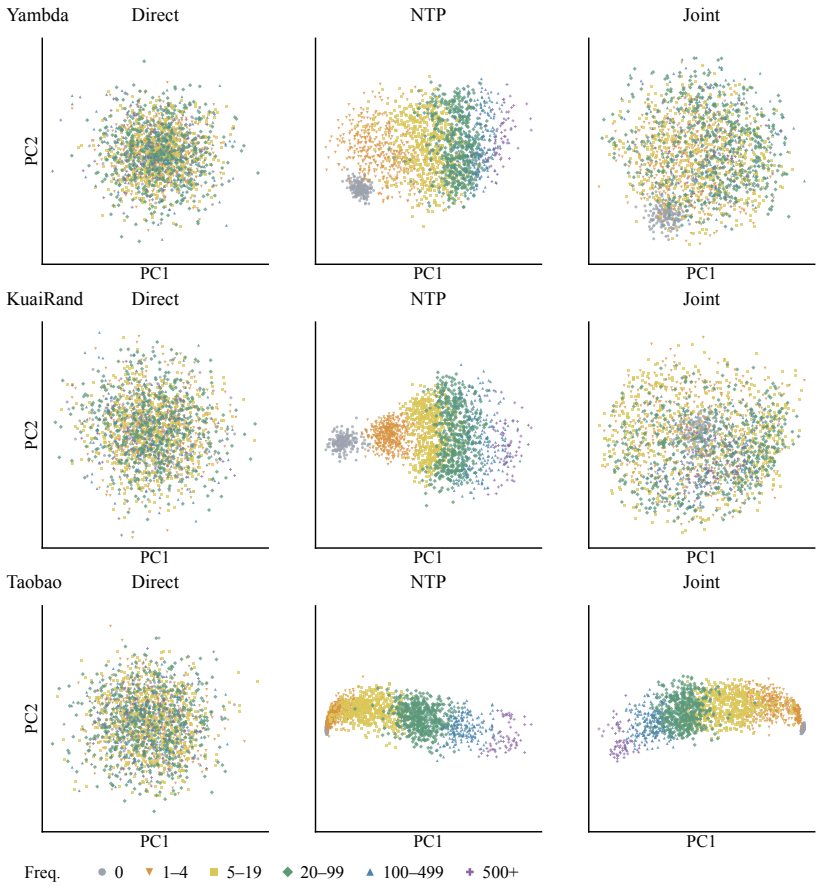


Figure 5: Frequency-colored PCA of item embeddings on Yambda, KuaiRand, and Taobao (rows). Columns compare Direct, NTP, and Joint (Cross+FA), using the same 1,966 items per dataset and six frequency groups.

In all three datasets, the frequency groups overlap substantially in Direct's two-dimensional projection. NTP's projection exhibits a clearer ordering along the first principal component: low-frequency items appear mainly toward the left and higher-frequency items toward the right. The pattern is visible in Yambda and Taobao and particularly pronounced in KuaiRand. Joint exhibits more overlap among nonzero-frequency groups on Yambda and KuaiRand, while a frequency ordering remains visible on Taobao. This plot describes the geometry of the saved embedding matrices; the transfer results are evaluated with the original embeddings, not their PCA projections.

C Accounting for Update Paths

We record the three gradient paths of the tied item table during the original NTP training run. Transparent autograd hooks on indexed lookups collect the history-input (C), observed positive-target output (P), and sampled negative-target output (N) gradients without changing the gradients passed to training. For each role, repeated occurrences of an item ID are added to the same row. The role buffers accumulate across the microbatches of an optimizer step and are divided by the training accumulation divisor. This gives step gradients g_k^P, g_k^C, and g_k^N; after each step, the buffers are reset. Their sum is checked against the actual tied-table data gradient before clipping.

Each role then receives the same global clipping factor c_k computed from the actual total gradient. We maintain a separate first-moment ledger for each role, while all three use the optimizer's one actual second moment v_k:

$$m_k^r = \beta_1 m_{k-1}^r + (1-\beta_1) c_k g_k^r, \tag{10}$$

$$a_k^r = -\eta_k \frac{m_k^r/(1-\beta_1^k)}{\sqrt{v_k/(1-\beta_2^k)}+\epsilon}, \qquad R_k^r = (1-\eta_k w_d) R_{k-1}^r + a_k^r, \quad r \in \{P, C, N\}. \tag{11}$$

Here a_k^r is the role's optimizer-step contribution, w_d is the actual decoupled weight-decay coefficient, and R_k^r accumulates the part of earlier role updates retained after subsequent decay. The ledger records, for tracked item rows, cumulative clipped role gradients and support counts, raw sums of role updates, and the final retained vectors R_T^P, R_T^C, and R_T^N. Decayed initialization and momentum or numerical residuals are recorded separately and used to check reconstruction of the trained table. Figure 2 scales these retained endpoint vectors, not raw gradient sums. This accounting follows the observed optimizer trajectory; removing a path and retraining would produce a different trajectory.

D Frequency Adjustment Experiments

This appendix supplements the frequency analysis in Section 4. The analyses below follow the experimental setup in Section 6: a one-layer, one-head Transformer with hidden dimension 128, effective batch size 512, maximum history length 512, and ten epochs per training stage. Results are reported over five random seeds. Each dataset uses an 8:1:1 training/validation/test split; downstream checkpoints are selected by validation AUC and evaluated on the held-out test set. The pretrained item table is transferred and frozen while the downstream model is trained. Appendix E gives the complete training and transfer settings.

Throughout, f_i counts occurrences of item i as an observed next-item target in the retained pretraining histories. Sampled negatives and downstream labels do not contribute to this count. NTP and FA use the same forward objective; FA scales only the positive-target output-embedding gradient. The counting rule and fixed multipliers are given in Equations (18)–(19). The frequency groups used to display an analysis do not change these multipliers.

D.1 Frequency-Conditioned Transfer Gains

Figure 3(a) compares Direct, NTP, and FA on Yambda, KuaiRand, and Taobao. Each comparison uses the same user, history, positive candidate, and negative candidate across methods. We compute the score margin and pairwise logistic loss defined in Equation (5). For each method and seed, one temperature is fitted on a disjoint calibration set and then held fixed across all frequency groups. Calibration changes the scale used to measure loss; it does not change the ordering of a method's scores.

Positive and negative candidates determine the row and column of each frequency cell. Table 5 makes the intervals in Figure 3(a) explicit; the tick labels in that figure indicate interval lower bounds. The displayed cells have nonzero target frequencies. KuaiRand uses different positive and negative intervals, so a cell's position must be interpreted using its two axis labels.

Table 5: Candidate-frequency intervals displayed in Figure 3(a). These intervals specify the analysis groups, not the FA multiplier profile.

Dataset	Positive candidate	Negative candidate
Yambda	1–4, 5–19, 20–99, 100–499, 500+	1–4, 5–19, 20–99, 100–499, 500+
KuaiRand	1–2, 3–6, 7–14, 15–39, 40+	1, 2, 3–4, 5–9, 10+
Taobao	1–4, 5–19, 20–99, 100–249, 250+	1–4, 5–19, 20–99, 100–249, 250+

Let $L_M(a, b)$ be the mean calibrated pairwise loss in a cell, as in Equation (6). In addition to NTP's gain G_{ab} over Direct, we measure FA's incremental gain by

$$\Delta G_{ab}^{\mathrm{FA}} = \frac{L_{\mathrm{NTP}}(a, b) - L_{\mathrm{FA}}(a, b)}{L_{\mathrm{Direct}}(a, b)}. \tag{12}$$

Both comparisons use the same Direct denominator, allowing the NTP gain and FA increment to be compared on a common scale. We form each loss ratio within a seed and then average the resulting gains across seeds. Figure 3(a) displays the uncentered quantities $100G_{ab}$ and $100\Delta G_{ab}^{\mathrm{FA}}$. A negative NTP gain means a higher loss than Direct; a negative FA increment means a higher loss than NTP. These are relative pairwise-loss changes, not changes in AUC.

D.2 Positive-Target Contributions by Frequency

Figure 3(b) examines how FA changes the retained positive-target component of the item embeddings. We use the update accounting in Appendix C, including each trajectory's actual clipping, optimizer moments, and weight decay. For method $M \in \{\mathrm{NTP}, \mathrm{FA}\}$, let $R_{T,i}^{P,M}$ denote the retained positive-target update for item i, and let $\Delta e_i^M = e_{T,i}^M - e_{0,i}^M$ be its displacement from initialization. For a frequency group B, the plotted normalized signed projection is

$$A_P^M(B) = 100\,\frac{\sum_{i \in B} \langle R_{T,i}^{P,M}, \Delta e_i^M \rangle}{\sum_{i \in B} \|\Delta e_i^M\|_2^2}. \tag{13}$$

The numerator and denominator are pooled over the items in the group before taking their ratio. Each method uses its own embedding displacement. This measures the component's signed contribution along that displacement; it is not a component norm or a fraction of downstream AUC. Contributions can exceed 100% or be negative when update components oppose one another.

All three datasets use the groups 1–4, 5–19, 20–99, 100–499, and 500+ in this panel. Items with $f_i = 0$ have no observed positive-target update and are omitted from the displayed curves. The shading and bars show $A_P^{\mathrm{FA}}(B) - A_P^{\mathrm{NTP}}(B)$ in percentage points, rather than confidence intervals. This measurement complements the downstream loss comparison by showing how positive-target contributions vary across frequency groups.

D.3 Relative Frequency and Transfer-Gain Regression

Figure 6 summarizes the association between relative candidate frequency and NTP's gain over Direct on Taobao, Yambda, KuaiRand, ContentWise, RecFlow, and Tenrec. The regression uses the absolute target-frequency intervals 0, 1–4, 5–19, 20–99, 100–499, and 500+. Each point represents a positive–negative frequency cell with at least 30 positive and 30 negative evaluation examples. This gives 36 retained cells each for Taobao, Yambda, ContentWise, and Tenrec, 24 for KuaiRand, and 12 for RecFlow.

For each dataset, let $\mathcal{B}$ be its set of retained cells. We summarize the frequency difference and center the relative loss reduction as

$$X_{ab} = \overline{\log(1+f)}^{+}_{ab} - \overline{\log(1+f)}^{-}_{ab},$$
$$\widetilde{G}_{ab} = G_{ab} - \frac{1}{|\mathcal{B}|} \sum_{(a',b') \in \mathcal{B}} G_{a'b'}. \tag{14}$$

The two means in X_{ab} are taken over the positive and negative evaluation examples in the cell, respectively, retaining repeated candidate occurrences. Thus $X_{ab} > 0$ means that positive candidates are more frequent on the log-count scale. The response uses the relative reduction in cell-average losses, rather than averaging individual loss ratios. Centering subtracts the unweighted mean benefit over retained cells; a negative $\widetilde{G}_{ab}$ therefore indicates a below-average gain, not necessarily a loss relative to Direct.

We fit a separate ordinary least-squares regression for each dataset d,

$$\widetilde{G}_{ab} = \alpha_d + \beta_d X_{ab} + \varepsilon_{ab}, \tag{15}$$

with an intercept and equal weight for each retained cell. Cells with more examples do not receive greater regression weight. Centering changes the intercept but leaves the slope and R^2 unchanged.

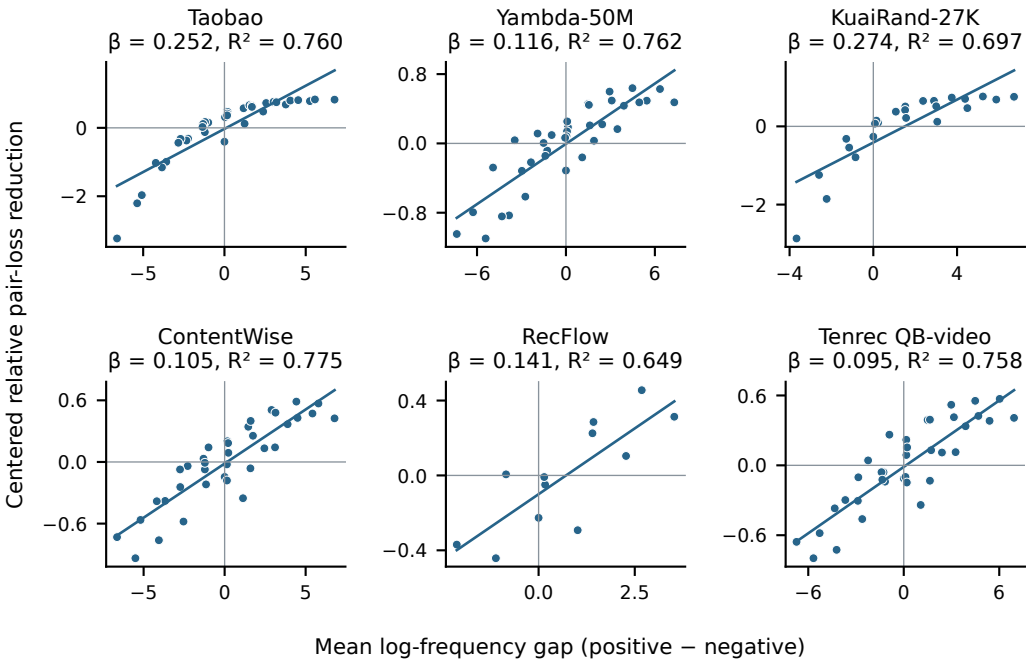


Figure 6: Relative candidate frequency and NTP transfer gains on six datasets. Each point is a retained frequency cell. The horizontal axis is the difference in mean log target counts X_{ab}; the vertical axis is the centered relative pairwise-loss reduction $\widetilde{G}_{ab}$. Lines are separate equal-cell linear fits with an intercept. Zero on the vertical axis denotes the dataset's mean retained-cell gain.

All six fitted slopes are positive: NTP's relative loss reduction tends to be larger when positive candidates are more frequent than negative candidates. This extends the relative-frequency pattern in Section 4.2 across datasets. The fitted slopes summarize this association at the level of frequency groups. FA's effect is evaluated by the NTP–FA comparison in Figure 3 and the downstream results in Table 1; the regression itself measures the NTP–Direct frequency pattern.

D.4 Exploratory Method Selection

As a preliminary design exploration, we compare two simple frequency-based alternatives to FA. Whole-sample weighting applies FA's target-frequency multiplier to the entire next-item loss. Post-hoc frequency balancing keeps NTP pretraining unchanged and subsequently adjusts the retained positive-target component to equalize its normalized signed projection across frequency groups. Each resulting item table is frozen during downstream training.

Table 6: Exploratory comparison of frequency-based alternatives against FA. Negative values indicate lower AUC than the corresponding FA reference.

Dataset	Whole-sample weighting Δ_{FA}	Post-hoc frequency balancing Δ_{FA}
Yambda	-0.026803	-0.035756
Taobao	-0.008204	-0.006263
KuaiRand	-0.008187	-0.016460

$\Delta_{\mathrm{FA}} = \mathrm{AUC}_m - \mathrm{AUC}_{\mathrm{FA}}$. Values are absolute AUC differences.

Both yield lower AUC than their corresponding FA references on all three datasets (Table 6). We therefore retained FA rather than these alternatives.

E Experimental Settings and Implementation Details

This appendix specifies the training configuration, candidate construction, objectives, transfer procedure, and evaluation protocol for the Transformer comparisons in Table 1. NTP, Cross, FA, and Cross+FA use the same next-item targets and downstream training procedure. Cross changes the pretraining encoder, FA changes the positive-target backward contribution, and Cross+FA combines these changes; the tables also denote this combination by Joint. Direct trains the downstream model from random initialization. DeepFM, DIN, and CAN use their respective architectures, as described in Section 6.

E.1 Shared Model and Training Configuration

Table 7 summarizes the common settings. The main Transformer comparisons use item-ID embeddings, with category and segment embeddings disabled. The pretraining input and output lookups share the same item table. The history encoder uses causal attention, rotary position embeddings, pre-RMSNorm, and residual attention and feed-forward blocks. Each applicable stage runs for ten epochs: Direct has one downstream stage, whereas the pretrained methods have a pretraining stage followed by downstream training.

E.2 Training Histories and Candidate Construction

Data partitions and histories. Each dataset is divided into training, validation, and test partitions in an 8:1:1 ratio, following Section 6. Compared methods use the same partitions and processed examples. Pretraining consumes observed item histories from the training partition. Within each history, events retain chronological order; out-of-vocabulary items are removed, the most recent 512 events are retained, and shorter histories are padded on the right. Histories with fewer than two retained events are discarded. Repeated item occurrences are preserved. ID 0 is reserved for padding and unknown items.

The downstream query item and binary click label are excluded from the pretraining input and target construction. An earlier occurrence of the query item's ID inside the observed history remains an ordinary history event. Following Section 3.1, a retained history is $S = (x_1, \ldots, x_L)$, and each prefix $S_{\le t}$ predicts the immediate next item x_{t+1}, for $1 \le t < L$. Thus a history contributes $L - 1$ prediction positions; padding and the final position without a next item contribute no loss.

Pretraining candidate pool. The negative-sampling pool is the configured set of unique, nonzero item IDs from training interactions. It is fixed before optimization and reused by all pretraining

Table 7: Shared settings for the main Transformer comparisons. PT and FT denote pretraining and downstream CTR training. The negative count is the number of draws before collision masking; the positive target is added separately.

Setting	PT	FT
Transformer layers / heads	1 / 1	1 / 1
Hidden / embedding size	128 / 128	128 / 128
Maximum history length	512	512
Effective batch size	512 sequences	512 labeled examples
Training budget	10 epochs	10 epochs
Optimizer	AdamW	AdamW
Adam parameters	$(\beta_1, \beta_2) = (0.9, 0.98)$; $\epsilon = 10^{-8}$	$(\beta_1, \beta_2) = (0.9, 0.98)$; $\epsilon = 10^{-8}$
Weight decay	0.1	0.1
Peak / minimum learning rate	5×10^{-3} / 5×10^{-4}	10^{-3} / 10^{-4}
Learning-rate decay / precision	Cosine / BF16	Cosine / BF16
Dropout / gradient norm cap	0 / 1.0	0 / 1.0
Training objective	Sampled next-item cross-entropy	Binary cross-entropy
Cosine-logit temperature	$\tau = 0.05$	Not applicable
Sampled negatives	4,096 draws per sequence	Stored labeled candidates
Item table	Trainable; tied input/output rows	Frozen after transfer; trainable for Direct

variants. Items appearing only in validation or test data do not enter this pool. Pool membership is distinct from the embedding vocabulary size V and from positive-target frequency f_i: an eligible training item can have zero occurrences as a next-item target after history filtering and truncation. The positive target x_{t+1} is taken directly from the retained history and scored separately, even if it is absent from the negative-sampling pool.

Negative draws, sharing, and collisions. On each pretraining forward pass, every sequence row draws 4,096 item IDs independently and uniformly with replacement from the training pool. That row's sampled multiset is shared across its valid prediction positions. At position t, draws equal to x_{t+1} are masked and are not redrawn. The remaining multiset is $\mathcal{N}_t$ in Section 3.1; its entries retain their sampled multiplicities. Other observed history items are eligible negatives. Consequently, 4,096 is the number of draws, not the number of distinct retained negative items. For example, if the draws include item IDs $(7, 9, 9)$ and the current target is item 7, both occurrences of item 9 contribute; at a position whose target is item 9, both occurrences are masked. Sampling is refreshed on subsequent forward passes.

Downstream candidates. Each downstream example is $((S, x), y)$, where S is the retained history, x is the candidate item, and $y \in \{0, 1\}$ is its binary label, as in Section 3.1. Positive-label sources and dataset-specific negative construction are listed in Table 3. These labeled candidates are constructed during preprocessing and fixed within each partition for all compared methods. They are separate from the fresh pretraining draws. For example, Taobao's sampled downstream negatives exclude items in the current sequence window, whereas KuaiRand excludes the current positive target but permits previously observed history items. Yambda uses an eligible explicit dislike when available and otherwise samples from its training catalog. Evaluation scores the stored labeled candidates; it does not perform full-catalog retrieval.

E.3 Objectives and Frozen-Embedding Transfer

Sampled next-item objective. We use the notation of Section 3.1: h_t is the predictive state for $S_{\le t}$, $e_i = E[i]$ is the tied item embedding, and the temperature-scaled cosine logit is

$$z_t(i) = \frac{1}{\tau} \left\langle \frac{h_t}{\|h_t\|}, \frac{e_i}{\|e_i\|} \right\rangle. \tag{16}$$

With $\mathcal{N}_t$ defined above, the target probability is

$$p_t(x_{t+1}) = \frac{\exp z_t(x_{t+1})}{\exp z_t(x_{t+1}) + \sum_{n \in \mathcal{N}_t} \exp z_t(n)}. \tag{17}$$

The sum over $\mathcal{N}_t$ counts every retained draw, so repeated item IDs contribute repeated terms. Training minimizes $\mathcal{L}_{\mathrm{NTP}}$ as defined in Equation (1), averaging the negative log-likelihood over valid prediction positions. No sampling-probability correction is applied to the logits. NTP, Cross, FA, and Cross+FA retain this forward objective and candidate-sampling rule.

Transfer and downstream loss. At the end of pretraining, only the final item-embedding table E is copied into a newly initialized downstream Transformer. The transferred table remains frozen for both history and candidate lookups; all remaining parameters ϕ, including the downstream encoder and classification head, are reinitialized and trained with $\mathcal{L}_{\mathrm{disc}}(E, \phi)$, as in Equation (2). The pretraining encoder, Cross query, prediction head, and optimizer state are not transferred. Direct instead learns E and ϕ jointly from random initialization under the same downstream budget.

The downstream Transformer appends the candidate to the retained history and uses the hidden state at the candidate position for binary classification. The 512-item history cap excludes this additional candidate token. Training minimizes binary cross-entropy on the stored labels, with no next-item auxiliary loss. Cross and FA therefore affect downstream predictions through the transferred embeddings, while all Transformer variants use the same downstream architecture and loss.

E.4 Cross and FA Implementation

Cross-attention encoder. Following Section 5, the first Cross layer applies pre-RMSNorm to the learned shared query b and item embeddings, projects them to Q/K/V, and applies RoPE to Q and K at their corresponding history positions. Its attention output is added to the incoming query, followed by a feed-forward block with pre-RMSNorm and a second residual addition. In deeper encoders, subsequent layers use causal self-attention over the previous layer's states, followed by the same feed-forward structure. Only the first layer reads the original item embeddings directly. At every layer, position t attends only to $j \le t$, excluding its next-item target x_{t+1}. A final RMSNorm and, when needed, an output projection produce the predictive state h_t. The initial query b is shared across users and positions; there is no additive position embedding or extra blank memory entry. All valid prefixes are processed in one forward pass. The one-layer main setting uses only the initial cross-attention layer.

Target-frequency counts. Following Section 4.1, f_i counts occurrences of item i as an observed next-item target. FA computes these counts once from the pretraining histories $S = (x_1, \ldots, x_L)$ in $\mathcal{D}$, using the notation of Section 3.1. Each history has already undergone the filtering and 512-item truncation above. The count is

$$f_i = \sum_{S \in \mathcal{D}} \sum_{t=1}^{L-1} \mathbf{1}[x_{t+1} = i]. \tag{18}$$

Each source row is counted once, and every retained target occurrence contributes one. Repeated IDs and overlapping source histories retain their occurrence multiplicity. Counts exclude the downstream query field, validation/test histories, and sampled negative draws, and are not multiplied by the number of training epochs.

Fixed multiplier profile. As specified in Section 4.1, the following profile is fixed a priori and shared across datasets, with no dataset-wise normalization or quantile binning:

$$r_i = r(f_i), \qquad r(f) = \begin{cases} 4, & 1 \le f \le 4, \\ 2, & 5 \le f \le 19, \\ 1.25, & 20 \le f \le 99, \\ 1, & f = 0 \text{ or } f \ge 100. \end{cases} \tag{19}$$

Row zero keeps unit scaling. The multiplier acts only on the positive-target output-embedding gradient, as in Equation (4). It leaves the forward logits and loss, the gradient sent to the predictive state, and the history-input and negative-target lookup gradients unchanged at a fixed forward state. Cross+FA applies this same rule to the Cross encoder's predictions.

E.5 Checkpoint Selection and Reporting

For each run, training uses the training partition, and downstream checkpoint selection uses validation ROC AUC. Following Section 6, we select the downstream checkpoint with the highest validation AUC and report its held-out test AUC as Best AUC. Test scores do not determine the selected checkpoint. AUC is computed over all eligible labeled examples in the corresponding partition. The main comparisons report the mean and sample standard deviation over five random seeds. Pretraining always transfers its final item table E; validation selection applies to downstream checkpoints. Diagnostic experiments that fix a particular seed, endpoint, or history subset specify those choices in their respective appendices.

F Hidden-State Sensitivity

This appendix details the two single-position perturbation experiments in Section 5 and Figure 4.

Models and predictive states. We compare NTP and Cross under the base Transformer configuration of Section 6: one layer, one attention head, hidden dimension 128, and a maximum history length of 512. For an observed prefix $S_{\le t} = (x_1, \dots, x_t)$, h is the final normalized state at the last visible position, used to predict $y = x_{t+1}$. NTP initializes the query and residual stream from the item embedding at that position. Cross instead uses the learned vector b, shared across users and positions, to attend to the observed item embeddings. The latest item remains visible to Cross through attention, as specified in Equation (8).

Single-position perturbation and response. Model parameters and the item table are fixed during measurement. Each trial begins with the original full input embedding sequence $\mathbf{e} = (e_{x_1}, \dots, e_{x_t})$ and adds a perturbation only to the embedding at position j:

$$\mathbf{e}_\alpha^{(j)} = (e_{x_1}, \dots, e_{x_j} + \boldsymbol{\delta}_j, \dots, e_{x_t}), \qquad \|\boldsymbol{\delta}_j\|_2 = \alpha r.$$

For each dataset, r is the median current-item embedding norm at NTP initialization. This reference scale is held fixed across methods, positions, and perturbation strengths within that dataset; equal α therefore gives equal absolute perturbation norms. The perturbation acts on one input occurrence after embedding lookup, leaving other occurrences and the stored item table unchanged. Before evaluating another position or strength, we restore the original input.

For each model, we compare its perturbed state $h_\alpha^{(j)}$ with its own unperturbed state h using $D_j(\alpha) = 1 - \cos(h, h_\alpha^{(j)})$, as defined in Equation (9). A larger value indicates a greater change in the predictive state's direction. The full visible history remains available in every trial.

Position scan: Figure 4(a). On KuaiRand, we fix $\alpha = 0.1$ and perturb each of the last 32 visible positions separately. Lag ℓ indexes position $j = t - \ell$, with $\ell = 0, \dots, 31$. NTP responds much more strongly at the last position than at the preceding 31 positions. Cross reduces this last-position response and shows less variation across positions. Each plotted response corresponds to a separate single-position intervention.

Perturbation-strength scan: Figure 4(b1)–(b3). On KuaiRand, Yambda, and Taobao, we vary α over the strengths shown in the figure and separately perturb x_t and x_{t-1}. At a given strength, both positions use the same norm αr. NTP is more sensitive at the last position on all three datasets. Cross markedly reduces the last-position response on KuaiRand and Yambda; the difference between the two models is relatively small on Taobao.

Connection to positive-target supervision. Equation (7) relates the positive-target gradient direction to the normalized predictive state. The two experiments measure the sensitivity of that state to individual history inputs. Downstream transfer performance is evaluated separately in Section 6.

G FINE-TUNING TRAJECTORIES AND OVERFITTING

In the main results, Best AUC is the held-out test AUC of the downstream checkpoint selected by validation AUC, as specified in Appendix E. Figure 7 separately illustrates evaluation trajectories for Yambda-50M, Taobao, and KuaiRand-27K; its stars mark maxima of the displayed diagnostic curves rather than define the main comparison's checkpoint-selection rule. Direct and NTP come from the primary five-method campaign. The third curve comes from a separately audited four-horizon CrossSoftNTP+FA campaign, which reuses the same downstream data and frozen-transfer configuration. It is a trajectory diagnostic, not the main table's hard next-item Joint run.

The trajectories show a loss-stability gap on these evaluation rows. Direct exhibits the strongest late-stage evaluation-loss divergence, accompanied by a pronounced ranking collapse on Taobao and KuaiRand. NTP retains substantially more of its ranking performance and limits the BCE increase, while the CrossSoftNTP+FA diagnostic is the most stable of the three curves.

A mechanical compute match would continue training Direct for another pretraining-length budget even after its evaluation curve has begun to deteriorate. That comparison measures the consequence of extending this trajectory rather than whether pretraining produces a better representation. Figure 7 aligns the ten-epoch downstream FT budgets. Direct receives only downstream CTR training, whereas NTP and the CrossSoftNTP+FA diagnostic also consume training histories in a separate ten-epoch PT stage. Total compute and training-data access therefore differ across methods.

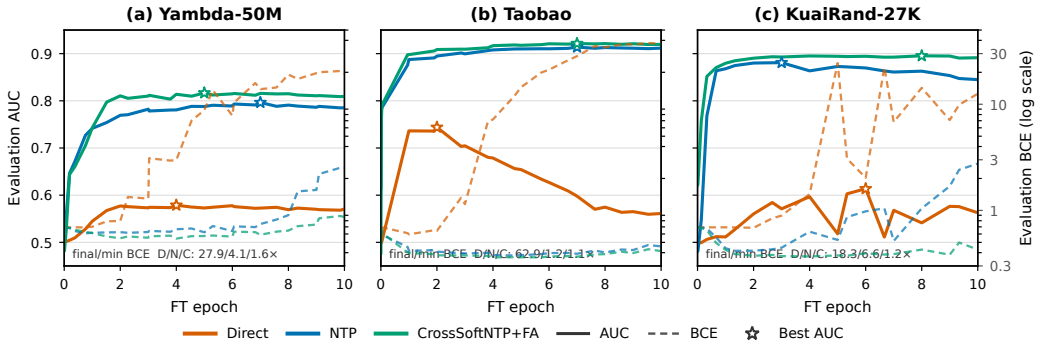


Figure 7: Downstream fine-tuning stability at seed 17. Each column shows one dataset; solid lines show evaluation AUC and dashed lines show hard-label evaluation BCE on a logarithmic scale. Steps are normalized to the shared ten-epoch FT budget, and stars mark each run's maximum logged evaluation AUC. The inset gives final/minimum BCE for Direct/NTP/CrossSoftNTP+FA (D/N/C). Direct and NTP come from the primary campaign; the third curve is a separate four-horizon diagnostic, not the main table's hard next-item Joint run. All displayed points are recorded evaluations without smoothing.