CREST: CALIBRATED LLM-BASED RECOMMENDATION WITH STATISTICAL PROMOTION CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM)-based recommender systems use item descriptions and interaction histories to personalize recommendations. However, attackers can manipulate this information to repeatedly promote selected items across users and rounds, displacing relevant alternatives and undermining trust in the platform. Existing defenses attempt to remove suspicious information but lack statistical guarantees on the promotion that remains. We therefore propose **C**alibrated LLM-based **RE**commendation with **ST**atistical Promotion Control (CREST), a target-agnostic framework for frozen LLM recommenders that controls each item's promotion rate within a predefined threshold at a user-specified confidence level while selecting top-K items per decision and limiting utility loss. Specifically, CREST first screens recommendation records by comparing their joint influence on rankings against historical effects and then uses sequence-level calibration to account for the promotion that remains after screening. This calibration guides a constrained selection procedure that favors highly ranked items while enforcing promotion thresholds without requiring knowledge of the attacker's targets. We further derive an exact characterization of worst-case promotion risk, enabling efficient optimization under the calibrated constraints. Under sequence-level exchangeability, we establish a finite-sample bound on the probability that the returned recommendations violate any candidate item's promotion threshold. Extensive experiments across multiple datasets, frozen LLM recommenders, and manipulation settings demonstrate the effectiveness and efficiency of CREST in controlling promotion while preserving recommendation utility. Our code is available at `https://anonymous.4open.science/r/CREST-8F90/`.

1 INTRODUCTION

Large language model (LLM)-based recommender systems can effectively utilize side information such as item descriptions alongside user interaction histories to better match individual preferences (Geng et al., 2022; Hou et al., 2024; Liao et al., 2024). However, this information itself, which supports user personalization, is open to manipulation. Recent studies show that attackers can utilize such manipulated information to promote selected items in recommendations (Zhang et al., 2024a; Wang et al., 2025). Since these edits can be semantically plausible, manipulated descriptions may be difficult to distinguish from genuine item information (Zhang et al., 2024a). As illustrated in Figure 1, a plausible edited description can promote an everyday sneaker even to a user seeking footwear for wet, rocky trails. Such promotion displaces alternatives better suited to users' needs, undermining the recommender's objective of matching items to individual preferences. Plausible edits that escape detection can continue to cause these displacements whenever the same descriptions are reused. The concern, therefore, extends beyond identifying manipulated records to controlling the item promotion they induce across repeated recommendations.

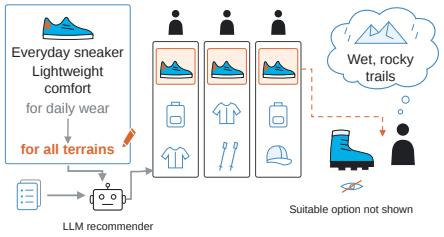


Figure 1: A plausible description edit can repeatedly promote an attacker-selected item, leaving alternatives better suited to users' needs unseen.

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.

In recent years, researchers have tried to develop several approaches to defend against such poisoning and instruction attacks. User reweighting (Zhang et al., 2024b) reduces the influence of poisoned interactions, while defensive fine-tuning strengthens instruction separation (Chen et al., 2025a). Both require changes during training and, therefore, do not directly apply to frozen or proprietary LLMs without fine-tuning access. Inference-time defense methods use textual or collaborative signals to identify and purify suspicious recommendation information (Wang et al., 2025; Ning et al., 2026). However, these methods determine the extent of purification heuristically, without statistically controlling the item promotion that remains across repeated recommendations. Parallely, recently developed calibration-based methods guide recommendation selection by accounting for uncertainty in item relevance (Angelopoulos et al., 2023). Yet, their guarantees do not bound manipulation-induced item promotion. These limitations motivate us to ask:

> *Given a frozen LLM recommender where the manipulated information and target items are unknown, can we bound each item's promotion rate by a predefined tolerance with a user-specified confidence level while preserving recommendation utility?*

To answer this question, we propose **C**alibrated LLM-based **RE**commendation with **ST**atistical Promotion Control (CREST), a target- and backbone-agnostic framework for frozen LLM recommenders that statistically controls item promotion across repeated recommendations. However, developing such a framework presents several key challenges. First, without manipulation labels, the influence of recommendation information is difficult to assess because its effect depends on the surrounding context, and removing isolated text fragments can distort the input's meaning. Second, repeated recommendations may be legitimate and do not by themselves reveal whether an item would have been recommended without manipulation, making promotion risk difficult to quantify. Most importantly, it remains to be seen how to translate this uncertainty in item-wise promotion rates into an intervention that satisfies the predefined promotion threshold while limiting degradation in recommendation utility?

CREST addresses these challenges by representing each recommendation record as a *RecAtom*, whose boundaries and identity are preserved in the prompt through Recommendation Record Markup (RRM). This allows the same record to be omitted in full wherever it appears. It then compares the rank deterioration caused by joint record omissions with a historical influence baseline, selecting combinations based on their excess influence and omission cost. Next, sequence-level calibration uses held-out clean–manipulated pairs to characterize the promotion remaining after screening, resulting in a family of plausible clean recommendation sequences. Finally, CREST selects exactly K items per decision under calibrated promotion constraints, favoring candidates ranked higher after screening. To make this optimization tractable, we derive an exact expression for the worst-case promotion rate over the calibrated family. We further establish finite-sample guarantees for promotion control simultaneously across all candidate items. The proposed framework is depicted in Figure 2. Our main contributions are summarized as follows:

- We formulate repeated item promotion across users and recommendation rounds as a target-agnostic risk-control problem for frozen LLM recommenders.
- To solve it, we develop CREST, which screens recommendation records by comparing joint omission effects with historical influence, then uses sequence-level calibration to guide top-K recommendations under the predefined promotion threshold.
- Theoretically, we establish finite-sample promotion control simultaneously across all candidate items (Theorem 1). We further derive an exact characterization of worst-case promotion rates (Lemma 1) and an efficient exact solution to the calibrated selection problem (Appendix D).
- Finally, we conduct comprehensive experiments across multiple datasets, frozen LLM recommenders, and manipulation settings, demonstrating CREST's effectiveness of its final selection, with guaranteed promotion-control consistent with our theoretical analysis.

2 Problem Formulation

We first introduce the notations used in the paper. Let $\mathcal{U}$ and $\mathcal{I}$ denote sets of users and items, respectively. Consider a sequence of T recommendation decisions made by a frozen LLM recommender f_θ such that at every $t \in \{1, \dots, T\}$, a prompt P_t is constructed for user $u_t \in \mathcal{U}$ and candidate set $\mathcal{I}_t \subseteq \mathcal{I}$ using recommendation data $\mathcal{D}_t$, where $|\mathcal{I}_t| \geq K$. The resulting Top-K recommendation is then given as $\mathcal{R}_t = f_\theta(P_t)$, with $\mathcal{R}_t \subseteq \mathcal{I}_t$ and $|\mathcal{R}_t| = K$. The recommendation data $\mathcal{D}_t$, however, may be manipulated to promote items that would otherwise be absent from users' Top-K recommendation list.

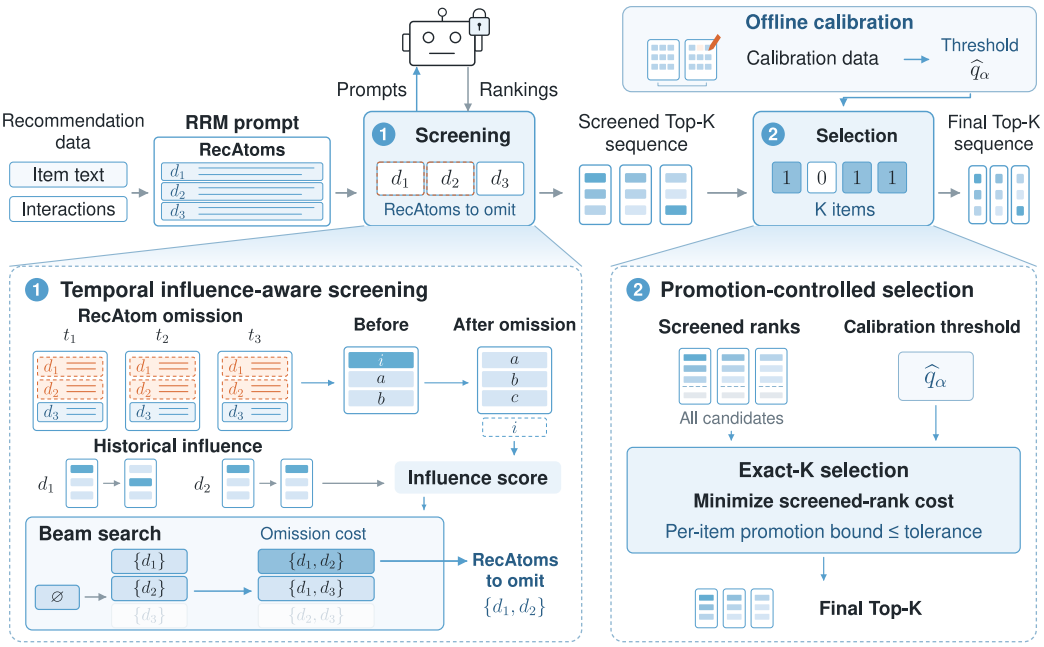


Figure 2: Overview of CREST. Temporal influence-aware screening compares joint RecAtom omission effects with historical influence. Offline sequence-level calibration guides exact-K selection under the predefined promotion threshold.

To measure this promotion, let $\mathcal{D}_t^*$ be the corresponding clean data and P_t^* prompt constructed from these data giving clean Top-K recommendation as $\mathcal{R}_t^* = f_\theta(P_t^*)$.

We quantify promotion across T decisions by defining an item-wise promotion rate for observed recommendation sequence $\mathcal{R}_{1:T}$ relative to its clean counterpart $\mathcal{R}_{1:T}^*$. Specifically, for each item i in $\mathcal{I}_{1:T} = \bigcup_{t=1}^{T} \mathcal{I}_t$, the Top-$K$ promotion rate is given as:

$$G_i(\mathcal{R}_{1:T}, \mathcal{R}_{1:T}^*) = \frac{1}{T} \sum_{t=1}^{T} \mathbb{I}\left[i \in \mathcal{R}_t \setminus \mathcal{R}_t^*\right]. \tag{1}$$

Here $G_i(.)$ measures how frequently item i is recommended when it would be absent from the corresponding clean Top-K recommendations. However, the clean recommendations for the evaluated sequence are unavailable when selecting the final recommendations, so its exact promotion rate cannot be computed directly. The manipulated information and promoted items are also unknown.

Under this uncertainty, we seek to limit item promotion while retaining higher-ranked recommendations. Let $r_{t,i} \in \{1, \ldots, K+1\}$ denote rank of candidate item i in recommender's output for user u_t, with $r_{t,i} = K+1$ for $i \in \mathcal{I}_t \setminus \mathcal{R}_t$. Given a predefined promotion threshold $\eta \in [0, 1]$ and calibration tolerance $\alpha \in (0, 1)$, we select exact-K recommendations $\widehat{\mathcal{R}}_{1:T} = (\widehat{\mathcal{R}}_1, \ldots, \widehat{\mathcal{R}}_T)$ by solving:

$$\min_{\widehat{\mathcal{R}}_t \in \binom{\mathcal{I}_t}{K},\ t=1,\ldots,T} \sum_{t=1}^{T} \sum_{i \in \widehat{\mathcal{R}}_t} r_{t,i} \quad \text{s.t.} \quad \Pr\left[\max_{i \in \mathcal{I}_{1:T}} G_i\left(\widehat{\mathcal{R}}_{1:T}, \mathcal{R}_{1:T}^*\right) \leq \eta\right] \geq 1 - \alpha. \tag{2}$$

3 The Proposed Framework

To control the item promotion without prior knowledge of the manipulated records or promoted items, we therefore design CREST, a backbone-agnostic framework for frozen LLM recommenders.

3.1 Recommendation Record Screening and Calibration

To begin with, we first screen and remove potentially manipulated recommendation data that exert unusual influence on an item's Top-K position. However, once recommendation data is flattened into an LLM prompt, removing arbitrary token spans can fragment the data, altering their meaning or breaking their association with the corresponding user or item. To address this, CREST changes the basic unit of screening itself, from token spans to recommendation records, namely *RecAtom*. It keeps the record's fields together and preserves its identity across recommendation decisions. At decision t,

we represent the recommendation data $\mathcal{D}_t$ as a collection of RecAtoms, with $d \in \mathcal{D}_t$ identifying an individual RecAtom. Its type τ_d specifies q_{τ_d} ordered fields, with $\kappa_{\tau_d,j}$ and $x_{t,d}^{(j)}$ denoting the name and value of its j-th field, respectively. Formally:

Definition 1. *A* **RecAtom** $d \in \mathcal{D}_t$ *is a typed recommendation record represented at decision* t *by* $\big(d, \tau_d, (\kappa_{\tau_d,j}, x_{t,d}^{(j)})_{j=1}^{q_{\tau_d}}\big)$. *It may appear in multiple recommendation decisions with decision-specific field values* $x_{t,d}^{(j)}$. *Omitting* d *removes the complete record from every decision in which it appears.*

To carry out these omissions, each RecAtom must be identifiable and removable as a complete record from the prompt P_t. We therefore introduce *Recommendation Record Markup* (RRM), a deterministic serialization (illustrated in Figure 3) that maps each RecAtom to one separately addressable `record` element containing its identifier, type, and ordered fields. Each `record` element encodes one RecAtom. We can therefore remove selected elements to assess the joint influence of the corresponding RecAtoms on item rankings, as described next.[1]

```
<recommendation_data>
  <record id="d" type="τ_d">
    <content>
      <field name="κ_{τ_d,1}"> x_{t,d}^{(1)} </field>
        ⋮
          <field name="κ_{τ_d,q_{τ_d}}">  x_{t,d}^{(q_{τ_d})}
</field>
    </content>
  </record>
    ⋮
</recommendation_data>
```

Figure 3: Recommendation Record Markup (RRM) for a RecAtom d at decision t.

Temporal influence-aware RecAtom screening. To screen for manipulation-induced item promotion, we score each RecAtom subset $B \subseteq \bigcup_{t=1}^{T} \mathcal{D}_t$ by how much the rank deterioration caused by its omission exceeds a temporal influence baseline:

$$\psi(B) = \max_{i \in \mathcal{I}_{1:T}} \frac{1}{T} \sum_{t=1}^{T} \max\{\delta_{t,i}(B) - \mu_{t,i}(B),\, 0\}\,. \tag{3}$$

Here, $\delta_{t,i}(B)$ measures the normalized rank deterioration of item i after omitting B at decision t, and $\mu_{t,i}(B)$ denotes the corresponding temporal influence baseline. The score averages positive excess influence across recommendation decisions and retains the largest item-wise response.

To compute $\delta_{t,i}(B)$, we construct the diagnostic prompt $P_t(B)$ by removing from P_t the `record` elements corresponding to $B \cap \mathcal{D}_t$. The resulting ordered Top-K recommendation is $\mathcal{R}_t^{(B)} = f_\theta(P_t(B))$, with $r_{t,i}^{(B)} \in \{1, \ldots, K\}$ denoting the rank of $i \in \mathcal{R}_t^{(B)}$. For each $i \in \mathcal{I}_{1:T}$, we define its rank deterioration relative to the observed recommendation $\mathcal{R}_t$ as:

$$\delta_{t,i}(B) = \begin{cases} \dfrac{\max\Big\{r_{t,i}^{(B)} - r_{t,i},\, 0\Big\}}{K + 1 - r_{t,i}}, & i \in \mathcal{R}_t \cap \mathcal{R}_t^{(B)}, \\ 1, & i \in \mathcal{R}_t \setminus \mathcal{R}_t^{(B)}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

To construct the temporal influence baseline $\mu_{t,i}(B)$, we first estimate the historical influence of each RecAtom on item i. For $d \in \mathcal{D}_t$, let $\mathcal{H}_{t,d,i} = \{s < t : d \in \mathcal{D}_s,\ i \in \mathcal{R}_s\}$ denote the preceding recommendation decisions in which RecAtom d was present and item i was recommended. Using temporal weights $w_{t,s} = \exp[-\beta(t-s)]$, where $\beta \geq 0$, we define:

$$\mu_{t,d,i} = \begin{cases} \dfrac{\sum_{s \in \mathcal{H}_{t,d,i}} w_{t,s}\, \delta_{s,i}(\{d\})}{\sum_{s \in \mathcal{H}_{t,d,i}} w_{t,s}}, & \mathcal{H}_{t,d,i} \neq \varnothing, \\ 0, & \mathcal{H}_{t,d,i} = \varnothing. \end{cases} \tag{5}$$

For joint omissions, we obtain the temporal influence baseline by aggregating and capping the corresponding singleton baselines:

$$\mu_{t,i}(B) = \min\left\{1,\ \sum_{d \in B \cap \mathcal{D}_t} \mu_{t,d,i}\right\}. \tag{6}$$

[1]The complete RRM grammar, ordering convention, escaping rules, and prompt construction details are provided in Appendix B.

RecAtom selection and sequence-level calibration. Evaluating $\psi(B)$ over all $2^{|\bigcup_{t=1}^{T}\mathcal{D}_t|}$ subsets is computationally impractical. To address this, we introduce an excess-influence-guided RecAtom beam search. For a penalty coefficient $\lambda \geq 0$, the selected subset is given by:

$$\widehat{B} \in \arg\max_{B\in\mathcal{Q}} \{\psi(B) - \lambda c(B)\}. \tag{7}$$

Here, $\mathcal{Q}$ denotes the subsets evaluated during search, including $\varnothing$, and $c(B) = T^{-1}\sum_{t:\,|\mathcal{D}_t|>0} |B \cap \mathcal{D}_t|/|\mathcal{D}_t|$ is the mean omission fraction across recommendation decisions. Starting from $\varnothing$, the search first evaluates all singleton omissions, whose rank deterioration scores are used to construct the temporal singleton baselines in Equation (5). It then expands each retained subset by one RecAtom and evaluates the resulting subsets. At each depth, the beam search retains the subsets with the largest penalized score $\psi(B) - \lambda c(B)$.

Applying $\widehat{B}$ gives the screened Top-K recommendation sequence:[2]

$$\overline{\mathcal{R}}_{1:T} := \left(\overline{\mathcal{R}}_1, \ldots, \overline{\mathcal{R}}_T\right), \qquad \overline{\mathcal{R}}_t := \mathcal{R}_t^{(\widehat{B})}.$$

However, these screened recommendations may still contain manipulation-induced item promotion. To bound this remaining promotion, we compare the screened recommendations with their clean references across N calibration sequences (Angelopoulos & Bates, 2023). For sequence $\ell \in \{1, \ldots, N\}$, we denote the screened and reference recommendation sequences by $\overline{\mathcal{R}}_{1:T}^{(\ell)}$ and $\mathcal{R}_{1:T}^{*,(\ell)}$, respectively. Using the largest item-wise promotion rate from each sequence, we obtain calibrated threshold as follows:

$$\widehat{q}_\alpha = \left[\left\{\max_{i\in\bigcup_{t=1}^{T}\mathcal{I}_t^{(\ell)}} G_i\left(\overline{\mathcal{R}}_{1:T}^{(\ell)}, \mathcal{R}_{1:T}^{*,(\ell)}\right)\right\}_{\ell=1}^{N}, 1\right]_{(\lceil (N+1)(1-\alpha)\rceil)}, \tag{8}$$

where $[\cdot]_{(k)}$ denotes the k-th order statistic in ascending order. For the screened sequence $\overline{\mathcal{R}}_{1:T}$, we use $\widehat{q}_\alpha$ to construct the following family of candidate reference Top-K sequences:

$$\mathfrak{C}_\alpha\left(\overline{\mathcal{R}}_{1:T}\right) = \left\{\mathcal{C}_{1:T} \in \prod_{t=1}^{T}\binom{\mathcal{I}_t}{K} \;\middle|\; \max_{i\in\mathcal{I}_{1:T}} G_i\left(\overline{\mathcal{R}}_{1:T}, \mathcal{C}_{1:T}\right) \leq \widehat{q}_\alpha\right\}. \tag{9}$$

Following proposition establishes a $1-\alpha$ coverage guarantee for reference Top-K sequence $\mathcal{R}_{1:T}^*$:

Proposition 1. *Suppose the largest item-wise promotion rates after screening are exchangeable across the N calibration sequences and the evaluated sequence. Then, for any predefined $\alpha \in (0,1)$:*

$$\Pr\left[\mathcal{R}_{1:T}^* \in \mathfrak{C}_\alpha\left(\overline{\mathcal{R}}_{1:T}\right)\right] \geq 1-\alpha.$$

Proof. See Appendix F.1. □

Remark 1. *Proposition 1 shows that the largest item-wise promotion rate remaining after screening is bounded by $\widehat{q}_\alpha$ with probability at least $1-\alpha$, establishing the finite-sample validity of* CREST*'s promotion calibration at inference without access to the reference sequence.*

3.2 Promotion-Controlled Top-K Recommendation

The remaining challenge is to translate this coverage guarantee into item-wise promotion control. To achieve that, we minimize total screened rank of selected items, subject to each item's promotion rate being at most η relative to every sequence in $\mathfrak{C}_\alpha(\overline{\mathcal{R}}_{1:T})$. Let $z_{t,i} \in \{0,1\}$ indicate whether item $i \in \mathcal{I}_t$ is included in the final recommendation at decision t. The set of exact-K selections is then:

$$\mathcal{Z}_K = \left\{\mathbf{z} : z_{t,i} \in \{0,1\}, \quad i \in \mathcal{I}_t, \qquad \sum_{i\in\mathcal{I}_t} z_{t,i} = K, \quad t = 1, \ldots, T\right\}.$$

To enforce the promotion threshold for a selection $\mathbf{z} \in \mathcal{Z}_K$, we define the largest promotion rate of each item $i \in \mathcal{I}_{1:T}$ over the calibrated reference family as:

$$\widehat{G}_i(\mathbf{z}) = \max_{\mathcal{C}_{1:T}\in\mathfrak{C}_\alpha(\overline{\mathcal{R}}_{1:T})} \frac{1}{T}\sum_{t:\,i\in\mathcal{I}_t} z_{t,i}\mathbb{I}[i \notin \mathcal{C}_t]. \tag{10}$$

We express the maximum in Equation (10) using two item-wise counts. Specifically, we define $a_i(\mathbf{z}) = \sum_{t:\,i\in\mathcal{I}_t\setminus\overline{\mathcal{R}}_t} z_{t,i}$ as the number of decisions at which $\mathbf{z}$ selects i outside the screened Top-K, and $b_i(\mathbf{z}) = \sum_{t:\,i\in\overline{\mathcal{R}}_t,\,|\mathcal{I}_t|>K} z_{t,i}$ as the number of decisions at which it selects i within the screened

[2]The screening procedure is detailed in Algorithm 1 in Appendix E.

Top-K and an alternative candidate is available. The following lemma gives an exact expression for the largest promotion rate using these counts, allowing us to enforce the threshold η for each item:

Lemma 1. *Let $h_\alpha = \lfloor T\widehat{q}_\alpha \rfloor$ denote the calibrated limit on exclusions of each item from the screened recommendation sequence. Given any exact-K selection $\mathbf{z} \in \mathcal{Z}_K$, the largest promotion rate of each item $i \in \mathcal{I}_{1:T}$ over the calibrated reference family can be expressed as:*

$$\widehat{G}_i(\mathbf{z}) = \frac{a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\}}{T}. \tag{11}$$

Proof. See Appendix F.2. □

Remark 2. *Lemma 1 reduces promotion control over the calibrated reference family to explicit item-wise constraints. This reformulation preserves the set of feasible exact-K selections.*

Finally, we combine the exact item-wise promotion evaluation in Lemma 1 with screened ranks to obtain a calibrated counterpart to Equation (2). Let $\bar{r}_{t,i}$ denote the screened rank of candidate $i \in \mathcal{I}_t$ following the defined rank convention. The final selection is then:

$$\widehat{\mathbf{z}} \in \arg\min_{\mathbf{z}\in\mathcal{Z}_K} \sum_{t=1}^{T} \sum_{i\in\mathcal{I}_t} \bar{r}_{t,i} z_{t,i} \quad \text{s.t.} \quad a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\} \le \lfloor T\eta \rfloor, \quad \forall i \in \mathcal{I}_{1:T}. \tag{12}$$

The final Top-K recommendation at decision t therefore given as:

$$\widehat{\mathcal{R}}_t := \{i \in \mathcal{I}_t : \widehat{z}_{t,i} = 1\}.$$

Equation (12) can be solved using minimum-cost flow formulation. Details are in Appendix D.

To this end, we complete the designing of CREST. The complete CREST procedure is summarized in Algorithm 2 in Appendix E and case study is provided in Appendix C.

4 Theoretical Analysis

In this section, we establish the promotion-control guarantee for the final Top-K recommendations produced by CREST.

Theorem 1. *Under the conditions of Proposition 1, let $\eta \in [0, 1]$ be such that Equation* (12) *is feasible almost surely. Then, any feasible solution provides an exact-K recommendation sequence $\widehat{\mathcal{R}}_{1:T}$ satisfying:*

$$\Pr\left[\max_{i\in\mathcal{I}_{1:T}} G_i\left(\widehat{\mathcal{R}}_{1:T}, \mathcal{R}^*_{1:T}\right) \le \eta\right] \ge 1 - \alpha. \tag{13}$$

Proof. See Appendix F.3. □

Remark 3. *Theorem 1 establishes finite-sample promotion control as with probability at least $1 - \alpha$, no candidate item is promoted in more than an η fraction of the recommendation decisions. This validates* CREST*'s control of repeated item promotion under exact-K selection.*

5 Experiments

In this section, we conduct experiments to evaluate the effectiveness of CREST. We evaluate it through the following research questions: **RQ1.** How effectively does CREST suppress item promotion while preserving recommendation quality? **RQ2.** How does its effectiveness vary with manipulation intensity and recurrence? **RQ3.** Does CREST achieve the prescribed coverage and promotion control, and how do α and η affect recommendation quality and exact-K feasibility? **RQ4.** How do CREST's components and screening parameters affect performance, and what is the cost of final selection? Full experimental details and additional results are provided in Appendices G and H, with hyperparameter analysis in Appendix H.3.

5.1 Datasets and Experimental Settings

Datasets and recommenders. We evaluate Amazon Beauty (Ni et al., 2019), Steam (Kang & McAuley, 2018), and MovieLens-1M (Harper & Konstan, 2015) using item metadata and interaction histories. Main results use the frozen Qwen2.5-7B-Instruct (Yang et al., 2024); additional comparisons with Qwen3.5-9B (Qwen Team, 2026) and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) appear in Appendix H.1. Each sequence contains $T = 20$ decisions with $K = 5$ and $|\mathcal{I}_t| = 50$ candidates per decision. Within each setting, all methods use the same frozen recommender, users, candidate sets, and observed data $\mathcal{D}_t$. Dataset statistics and preprocessing are given in Appendix G.1.

Table 1: Comparison between CREST and baselines with Qwen2.5-7B-Instruct on Amazon Beauty, Steam, and MovieLens-1M under three attacks, averaged over five seeds (ten test sequences). R (Recall@5), N (NDCG@5), and FPR are in %; $G_{\max}$ is averaged across sequences. Bold/underline indicate best/second-best values per dataset–attack setting. – indicates methods that do not remove records. † marks $G_{\max} > \eta = 0.10$.

	Amazon Beauty				Steam				MovieLens-1M			
Method	R↑	N↑	$G_{\max}$ ↓	FPR↓	R↑	N↑	$G_{\max}$ ↓	FPR↓	R↑	N↑	$G_{\max}$ ↓	FPR↓
Instruction injection												
Backbone	11.00	6.50	0.140†	–	12.00	6.75	0.110†	–	12.00	6.92	0.165†	–
Spotlighting	9.50	4.96	0.180†	–	12.00	6.61	0.185†	–	9.00	4.93	0.230†	–
Prompt Guard 2–Record	11.50	6.14	0.175†	0.061	12.50	7.24	0.175†	0.071	7.50	4.53	0.205†	0.078
Random-Matched	9.50	5.46	0.150†	0.058	13.00	7.46	0.115†	0.072	12.00	6.71	0.165†	0.079
CREST	**13.50**	**7.38**	**0.090**	**0.046**	**14.50**	**8.11**	**0.080**	**0.063**	**13.50**	**7.29**	**0.095**	**0.075**
Deceptive text rewriting												
Backbone	11.50	7.04	0.140†	–	11.50	7.13	0.110†	–	7.50	4.45	0.110†	–
Spotlighting	12.50	7.66	0.135†	–	11.50	6.97	0.100	–	11.00	6.52	0.125†	–
Prompt Guard 2–Record	11.50	7.04	0.065	0.066	11.50	7.13	0.110†	0.068	7.50	4.45	0.090	0.065
RewriteDetection-Record	6.50	3.95	0.085	2.711	12.50	7.39	0.080	1.310	7.00	4.28	0.075	0.611
Random-Matched	12.50	7.72	0.115†	**0.061**	11.50	6.82	0.110†	0.070	11.50	6.82	0.070	0.066
CREST	**13.00**	**9.25**	**0.060**	**0.061**	**14.50**	**8.60**	**0.055**	**0.061**	**13.00**	**7.12**	**0.055**	**0.060**
Interaction-history manipulation												
Backbone	10.50	5.95	0.120†	–	11.00	6.38	0.115†	–	9.00	5.20	0.140†	–
Spotlighting	11.00	6.55	0.155†	–	12.00	6.86	0.155†	–	9.50	4.58	0.190†	–
Prompt Guard 2–Record	11.00	5.95	0.070	**0.077**	11.50	6.38	0.085	0.072	9.00	5.20	0.140†	0.997
RETURN-Del	12.50	7.00	0.085	2.407	12.00	7.23	0.090	2.420	9.50	5.91	**0.065**	1.740
Random-Matched	9.50	5.19	0.065	0.102	10.50	6.15	0.095	0.081	9.00	4.98	0.165†	0.111
CREST	**13.50**	**7.76**	**0.055**	0.081	**12.50**	**7.73**	**0.070**	**0.041**	**10.00**	**6.03**	0.080	**0.066**

Manipulation settings. We evaluate three types of manipulation. (a) Instruction injection appends target-promoting instructions to item-description fields. (b) Deceptive text rewriting adapts TextSimu (Wang et al., 2025) to alter candidate descriptions in favor of the target without explicit recommendation instructions. (c) Interaction-history manipulation inserts target-supporting interaction records, following RETURN's inference-time setting (Ning et al., 2026). Attack construction and budgets are detailed in Appendix G.2.

Baseline methods. We compare CREST with the unprotected Backbone, Spotlighting (Hines et al., 2024), and Prompt Guard 2–Record (Chennabasappa et al., 2025). For deceptive rewriting and history manipulation, we also evaluate record-removal adaptations of RewriteDetection (Wang et al., 2025) and RETURN (Ning et al., 2026), denoted by RewriteDetection-Record and RETURN-Del. Random-Matched randomly removes records while matching CREST's number of omitted identities and omission cost, without final selection. Baseline implementations are described in Appendix G.3.

Evaluation metrics. We measure recommendation quality using Recall@K and NDCG@K against held-out positive items (Järvelin & Kekäläinen, 2002), and promotion using:

$$G_{\max}(\mathcal{S}_{1:T}) = \max_{i \in \mathcal{I}_{1:T}} G_i(\mathcal{S}_{1:T}, \mathcal{R}^*_{1:T}), \tag{14}$$

where $\mathcal{S}_{1:T}$ is the sequence returned by a method. Recommendation quality and sequence-level $G_{\max}$ are averaged over test sequences returned. We also report the false-positive rate (FPR) of RecAtom removal and the exact-K return rate. Metric definitions are provided in Appendix G.5.

5.2 Experimental Results

5.2.1 Overall Effectiveness (RQ1)

Table 1 compares CREST with the baselines using Qwen2.5-7B-Instruct. On Amazon Beauty and Steam, CREST achieves the highest Recall@5 and NDCG@5 and the lowest mean $G_{\max}$ under all three attacks. Across all reported dataset–attack settings, its mean $G_{\max}$ remains below $\eta = 0.10$. These gains extend to deceptive rewriting and history manipulation, which don't rely on explicit malicious instructions. This is because CREST screens records by their influence on item rankings and constrains promotion during final selection. It also achieves higher recommendation quality and lower mean promotion than Random-Matched at the same omission cost, supporting the combination of influence-aware screening and constrained selection.

The attack-specific comparisons reveal a tradeoff between promotion reduction and unnecessary record removal. Under rewriting, CREST achieves higher recommendation quality, lower mean promotion, and lower FPR than RewriteDetection-Record on both Amazon Beauty and Steam. On MovieLens-1M under history manipulation, RETURN-Del achieves lower mean $G_{\max}$ (0.065 versus 0.080), whereas CREST achieves slightly higher recommendation quality and substantially lower

FPR (0.066% versus 1.740%). To further evaluate the LLM agnosticity, we conduct experiments with Llama-3.1-8B-Instruct and Qwen3.5-9B (Appendix H.1). For both backbones, CREST improves Recall@5 and NDCG@5 while reducing mean $G_{\max}$ relative to Backbone in all three settings.

These results demonstrate that CREST's benefits extend to newer LLMs, improving recommendation quality and promotion control across the diverse domains.

5.2.2 Results w.r.t. Attack Robustness (RQ2)

Having established CREST's effectiveness in RQ1, we next examine its robustness under different attack conditions using instruction injection on Amazon Beauty with Qwen2.5-7B-Instruct. Our analysis has two parts. First, we examine *manipulation intensity* by varying the number of corrupted records per affected prompt, and *recurrence* by varying how many decisions contain the same three corrupted records (Figure 4). Second, we examine *attack timing* by placing manipulation early, late, or dispersed throughout the sequence while keeping the corrupted records and number of affected decisions fixed. Due to space constraints, the attack-timing results are reported in Appendix H.2.

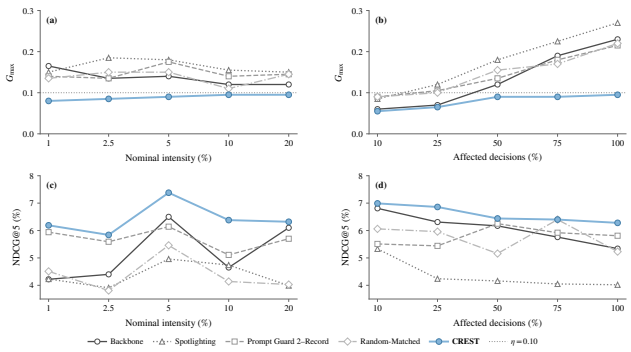


Figure 4: Robustness to manipulation intensity and recurrence on Amazon Beauty with Qwen2.5-7B-Instruct using NDCG@5 and $G_{\max}$ averaged over ten sequences across five seeds. The dotted line marks $\eta = 0.10$.

Figure 4 shows CREST maintains promotion control as manipulation intensity increases (mean $G_{\max}$ stays between 0.080 and 0.095) while every baseline exceeds $\eta = 0.10$ at each tested level. The gap to Random-Matched further shows that matching removal volume does not reproduce CREST's protection. CREST evaluates promotional influence of corrupted evidence and constrains resulting item exposure, addressing manipulation in both recommendation context and final outputs. The advantage becomes more pronounced as manipulation recurs. Every baseline exhibits increasing promotion, and the gap from CREST grows as same corrupted records appear in more decisions. CREST limits this accumulation while retaining its NDCG advantage. This pattern is consistent with its sequence-level selection, which imposes item-wise promotion constraints across decisions. Together, these results show that CREST achieves the highest or tied-highest NDCG@5 across all tested conditions while maintaining the lowest mean $G_{\max}$, consistently below the promotion threshold.

5.2.3 Results w.r.t. Statistical Promotion Control (RQ3)

Given a predefined promotion threshold η and calibration tolerance α, we examine whether CREST provides the specified promotion control while maintaining recommendation quality. We evaluate instruction injection on Amazon Beauty with Qwen2.5-7B-Instruct. We vary α at $\eta = 0.10$ to evaluate clean-family miscoverage, and vary η at $\alpha = 0.10$ to examine promotion control and recommendation quality. The experiments use 20 independent calibration blocks, each containing 19 calibration and five test sequences ($|\mathcal{I}_t| = 50$, $K = 5$, $T = 20$). Figure 5(a) shows that empirical clean-family

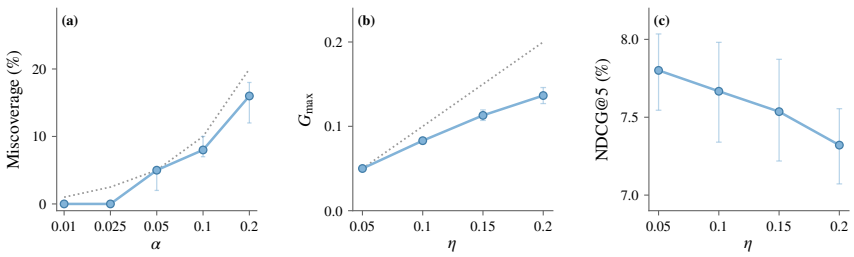


Figure 5: Effect of calibration α and promotion threshold η on Amazon Beauty with Qwen2.5-7B-Instruct. (a) Clean-family miscoverage versus α at $\eta = 0.10$; (b) mean $G_{\max}$ and (c) NDCG@5 versus η at $\alpha = 0.10$. Blue denotes CREST; whiskers show 95% confidence intervals; dotted lines mark nominal α in (a) and η in (b).

miscoverage remains at or below the specified calibration tolerance α. As α decreases, CREST constructs a larger calibrated reference family, reducing the frequency with which the clean reference

is excluded. Increasing α permits a smaller family and correspondingly higher miscoverage, showing that calibration responds to the selected tolerance.

Figure 5(b) shows that a smaller promotion threshold η leads to lower mean $G_{\max}$. CREST's final selection restricts repeated recommendations of the same item across the sequence, with stricter thresholds allowing less repeated exposure. The observed mean $G_{\max}$ remains at or below η across the feasible settings.

Figure 5(c) shows that stricter promotion control also accompanies higher observed recommendation quality. Mean NDCG@5 increases from 7.32% to 7.80% as η decreases from 0.20 to 0.05. Together with Figure 5(b), this indicates that allowing greater repeated item exposure provides no observed utility advantage in these settings. These results support promotion control at user-specified thresholds. Within the feasible range, system operators can select η and α according to their promotion-control requirements while retaining recommendation quality.

5.2.4 Results w.r.t. Ablation Analysis (RQ4)

Finally, we perform an ablation analysis to examine the contribution of record screening and final selection to CREST's recommendation performance and promotion control. We compare full CREST with *Without Screening*, *Screening-Only*, and *Random-Matched* under instruction injection on Amazon Beauty using Qwen2.5-7B-Instruct (Table 2). Without Screening retains final selection with its own calibration, whereas Screening-Only omits final selection. Random-Matched uses random omissions matched to CREST's removal budget, without final selection. All variants return complete recommendations.

Table 2: Component ablation on Amazon Beauty under instruction injection with Qwen2.5-7B-Instruct. Results use ten sequences across five seeds. Bold/underlining mark best/second-best values, including ties; † marks mean $G_{\max} > 0.10$.

Variant	R↑	N↑	$G_{\max}$ ↓	FPR↓
Without Screening	<u>11.50</u>	<u>6.73</u>	**0.080**	–
Screening-Only	10.00	6.00	0.145†	**0.046**
Random-Matched	9.50	5.46	0.150†	<u>0.058</u>
CREST	**13.50**	**7.38**	<u>0.090</u>	**0.046**

As shown in Table 2, full CREST achieves the highest Recall@5 and NDCG@5 while keeping mean $G_{\max}$ below $\eta = 0.10$. Removing final selection increases mean $G_{\max}$ from 0.090 to 0.145, exceeding the promotion threshold. Screening identifies influential evidence, but the remaining rankings can still repeatedly favor the same item. Final selection addresses this effect by constraining item-wise promotion across the sequence.

Removing screening leads to slightly lower mean promotion, but reduces both Recall@5 and NDCG@5. This suggests that screening helps preserve useful ranking information for final selection. Moreover, Screening-Only outperforms Random-Matched in recommendation quality and mean promotion under matched omission budgets, supporting the benefit of selecting records by their promotional influence. These results support the joint design of CREST: screening improves recommendation quality, while final selection constrains repeated item-wise promotion.

Inference efficiency. We compare CREST's final-stage recommendation time with the baselines on Amazon Beauty and Steam using Qwen2.5-7B-Instruct. As shown in Table 3, CREST achieves latency comparable to Backbone on both tested sequences. Recommendation generation, a step shared by all methods, accounts for over 99.8% of CREST's measured final-stage time; final selection requires only 18–19 ms per sequence. Further time and space analysis is performed in Appendix H.4. Together with the effectiveness results, this shows that CREST's final selection applies calibrated promotion constraints with minimal additional latency.

Table 3: Time efficiency comparison under instruction injection with Qwen2.5-7B-Instruct. Values are mean seconds per 20-decision sequence.

Method	Beauty	Steam
Backbone	17.582	15.775
Spotlighting	20.108	17.084
Prompt Guard 2–Record	17.779	15.805
Random-Matched	17.702	16.304
CREST	17.641	15.918

6 Conclusion

This paper investigates manipulation-induced item promotion across repeated decisions made by frozen LLM-based recommenders. We propose CREST, which combines temporal influence-aware RecAtom screening with sequence-level calibration to select exactly K items under a prescribed promotion threshold. Our analysis establishes finite-sample promotion control under the stated assumptions and an efficient exact solution to the final selection problem. Experiments demonstrate reduced item promotion while retaining recommendation utility across different manipulation settings. This work lays a foundation for reliable personalization in LLM-based recommender systems. Future work will extend CREST to multi-turn conversational recommendation and continual user-state adaptation.

AI USE STATEMENT

In this work, we used Qwen2.5-7B-Instruct to generate manipulated item descriptions for the deceptive text rewriting experiments in RQ1. Additionally, we used Grammarly tool to correct any grammatical errors and improve manuscript readability.

REPRODUCIBILITY STATEMENT

Implementation details and experimental settings are provided in Appendix G.4. The theoretical results are accompanied by proofs and remarks clarifying their assumptions and implications. The code repository provided includes data preprocessing procedures and scripts for running the experiments and generating tables and figures.

REFERENCES

Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows: theory, algorithms and applications*. Prentice hall, 1995.

Anastasios Angelopoulos, Stephen Bates, Adam Fisch, Lihua Lei, and Tal Schuster. Conformal risk control. In *International conference on learning representations*, volume 2024, pp. 55198–55218, 2024.

Anastasios N Angelopoulos and Stephen Bates. Conformal prediction: A gentle introduction. *Foundations and Trends in Machine Learning*, 16(4):494–591, 2023.

Anastasios N Angelopoulos, Karl Krauth, Stephen Bates, Yixin Wang, and Michael I Jordan. Recommendation systems with distribution-free reliability guarantees. In *Conformal and Probabilistic Prediction with Applications*, pp. 175–193. PMLR, 2023.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM conference on recommender systems*, pp. 1007–1014, 2023.

Stephen Bates, Anastasios Angelopoulos, Lihua Lei, Jitendra Malik, and Michael Jordan. Distribution-free, risk-controlling prediction sets. *Journal of the ACM (JACM)*, 68(6):1–34, 2021.

Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. {StruQ}: Defending against prompt injection with structured queries. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 2383–2400, 2025a.

Sizhe Chen, Arman Zharmagambetov, Saeed Mahloujifar, Kamalika Chaudhuri, David Wagner, and Chuan Guo. Secalign: Defending against prompt injection with preference optimization. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2833–2847, 2025b.

Sahana Chennabasappa, Cyrus Nikolaidis, Daniel Song, David Molnar, Stephanie Ding, Shengye Wan, Spencer Whitman, Lauren Deason, Nicholas Doucette, Abraham Montilla, et al. Llamafirewall: An open source guardrail system for building secure ai agents. *arXiv preprint arXiv:2505.03574*, 2025.

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.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. Defending against indirect prompt injection attacks with spotlighting. *arXiv preprint arXiv:2403.14720*, 2024.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. Large language models are zero-shot rankers for recommender systems. In *European conference on information retrieval*, pp. 364–381. Springer, 2024.

Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

Jinyuan Jia, Yupei Liu, Yuepeng Hu, and Neil Zhenqiang Gong. {PORE}: Provably robust recommender systems against data poisoning attacks. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 1703–1720, 2023.

Venkateswara Rao Kagita, Arun K Pujari, Vineet Padmanabhan, and Vikas Kumar. Inductive conformal recommender system. *Knowledge-Based Systems*, 250:109108, 2022.

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

Wonbin Kweon, Sanghwan Jang, SeongKu Kang, and Hwanjo Yu. Uncertainty quantification and decomposition for llm-based recommendation. In *Proceedings of the ACM on Web Conference 2025*, pp. 4889–4901, 2025.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 1258–1267, 2023.

Minghan Li, Xinyu Zhang, Ji Xin, Hongyang Zhang, and Jimmy Lin. Certified error control of candidate set pruning for two-stage relevance ranking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 333–345, 2022.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. Llara: Large language-recommendation assistant. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 1785–1795, 2024.

Jianghao Lin, Rong Shan, Chenxu Zhu, Kounianhua Du, Bo Chen, Shigang Quan, Ruiming Tang, Yong Yu, and Weinan Zhang. Rella: Retrieval-enhanced large language models for lifelong sequential behavior comprehension in recommendation. In *Proceedings of the ACM Web Conference 2024*, pp. 3497–3508, 2024.

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 (EMNLP-IJCNLP)*, pp. 188–197, 2019.

Liangbo Ning, Wenqi Fan, and Qing Li. Retrieval-augmented purifier for robust llm-empowered recommendation. *ACM Transactions on Information Systems*, 44(4):1–42, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL `https://qwen.ai/blog?id=qwen3.5`.

Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Representation learning with large language models for recommendation. In *Proceedings of the ACM web conference 2024*, pp. 3464–3475, 2024.

Zeyu Shen, Basileal Imana, Tong Wu, Chong Xiang, Prateek Mittal, and Aleksandra Korolova. Reliabilityrag: Effective and provably robust defense for rag-based web-search. *Advances in Neural Information Processing Systems*, 38:45662–45702, 2026.

Zongwei Wang, Min Gao, Junliang Yu, Xinyi Gao, Quoc Viet Hung Nguyen, Shazia Sadiq, and Hongzhi Yin. Id-free not risk-free: Llm-powered agents unveil risks in id-free recommender systems. In *Proceedings of the 48th international ACM SIGIR conference on research and development in information retrieval*, pp. 1902–1911, 2025.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. A survey on large language models for recommendation. *World Wide Web*, 27(5):60, 2024.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Jinghao Zhang, Yuting Liu, Qiang Liu, Shu Wu, Guibing Guo, and Liang Wang. Stealthy attack on large language model based recommendation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5839–5857, 2024a.

Kaike Zhang, Qi Cao, Yunfan Wu, Fei Sun, Huawei Shen, and Xueqi Cheng. Lorec: Combating poisons with large language model for robust sequential recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1733–1742, 2024b.

INDEX

- Related Work . . . . . p. 13
- Recommendation Record Markup . . . . . p. 13
- Case Study . . . . . p. 15
- Solution of the Optimization Problem . . . . . p. 15
- Algorithms . . . . . p. 18
- Proofs . . . . . p. 19
- Experimental Details . . . . . p. 20
- Additional Experimental Results . . . . . p. 24

A RELATED WORK

LLM-based recommendation and robustness. LLMs have enhanced recommendation through textual representation learning, instruction tuning, and prompting (Wu et al., 2024). Representative methods such as P5, Recformer, and TALLRec use language-based pretraining or recommendation-specific fine-tuning to model item information and user interactions (Geng et al., 2022; Li et al., 2023; Bao et al., 2023). LLaRA and RLMRec further integrate semantic and collaborative representations to capture complementary preference information (Liao et al., 2024; Ren et al., 2024). Other approaches directly prompt pretrained LLMs to rank candidate items from user histories (Hou et al., 2024), or retrieve relevant behaviors to improve the use of long interaction sequences (Lin et al., 2024).

Recent manipulation studies have shown that modifying item text can increase the exposure of selected items (Zhang et al., 2024a; Wang et al., 2025). Defenses address these vulnerabilities through various methods like training or input purification. LoRec reweights users during recommender training to reduce the influence of poisoned interactions (Zhang et al., 2024b). In the broader setting of prompt injection, StruQ and SecAlign strengthen resistance through structured instruction tuning and preference optimization, respectively (Chen et al., 2025a;b). These approaches require model updates. Inference-time defenses instead inspect recommendation information. RewriteDetection uses textual similarity and changes in recommendation frequency after rewriting (Wang et al., 2025), while RETURN uses collaborative signals to purify suspicious histories and aggregates the resulting predictions (Ning et al., 2026). Certified approaches provide guarantees under specified corruption models. PORE certifies recommendation overlap under bounded fake-user injection using recommenders trained on subsampled user data (Jia et al., 2023). ReliabilityRAG selects consistent retrieval evidence through contradiction checks, with robustness guarantees under bounded corruption and assumptions on contradiction-detection errors (Shen et al., 2026). These certificates concern recommendation overlap or the reliability of selected evidence.

Uncertainty and statistical calibration. Uncertainty estimation provides another perspective on recommendation reliability. Kweon et al. (2025) quantify predictive uncertainty in LLM-based recommendation and decompose it into recommendation uncertainty and prompt uncertainty. Risk-controlling prediction sets and conformal risk control provide general frameworks for calibrating prediction rules with finite-sample guarantees on specified losses (Bates et al., 2021; Angelopoulos et al., 2024). Recommendation-specific applications include inductive conformal methods that construct sets using calibrated nonconformity scores (Kagita et al., 2022) and recommendation selection with false discovery rate control (Angelopoulos et al., 2023). Related work in information retrieval calibrates candidate pruning to accelerate reranking while satisfying prescribed accuracy constraints (Li et al., 2022). These recommendation and ranking applications primarily address predictive reliability and relevance errors. Relevance alone, however, does not establish whether manipulated information repeatedly increases an item's exposure relative to unmanipulated recommendations.

B RECOMMENDATION RECORD MARKUP

RRM represents each RecAtom $d \in \mathcal{D}_t$ as one complete `record` element. We specify its grammar, record schemas, ordering convention, and escaping rules for the screening procedure.

Grammar and record schemas. The serialization follows the EBNF grammar below, where braces denote zero or more repetitions and `NL` is a line-feed character. Quoted strings are literal delimiters.

```
Data    ::= "<recommendation_data>" NL
            { Record NL } "</recommendation_data>"
Record  ::= Open NL { Field NL } "</content></record>"
Open    ::= '<record id="' Attr '" type="' Attr '"><content>'
```

Table 4: RecAtom schemas used in the experiments. Fields appear in the listed order; the bracketed rating field is included when supplied by the interaction construction.

Type	Trusted references	Ordered fields
`interaction`	User, item	`user_id`, `item_id`, `title`, `timestamp`, `event`, [`rating`]
`item_text`	Item	`item_id`, `title`, `description`

```
Field   ::= '<field name="' Name '">' Text '</field>'
Name    ::= Lower { Lower | Digit | "_" }
Attr    ::= EscChar { EscChar }
Text    ::= { EscChar }
EscChar ::= Literal | Entity
Entity  ::= "&amp;" | "&lt;" | "&gt;" | "&quot;" | "&#x27;"
```

Here, `Lower` and `Digit` denote ASCII lowercase letters and digits, respectively. `Literal` denotes any character other than `&`, <, >, `"`, or `'`; these five characters use the entity encodings above. Record identifiers, types, field names, and field order are supplied by the trusted recommendation-data pipeline. Record identifiers are nonempty and unique within each decision; field names are unique within each record. User and item references appear as `user_id` and `item_id` fields within `content`; their values remain trusted even though they share the field syntax with editable content.

Each interaction event forms one RecAtom. An item's title and description belong to the same `item_text` RecAtom and cannot be omitted separately. Original interaction identities derive from the dataset namespace and source-row key; item-text identities derive from the canonical item identifier. These identities remain fixed when content changes or a record reappears. New interaction records receive distinct identities through trusted ingestion and retain them in subsequent appearances.

Ordering and escaping. Interaction records appear first, in increasing timestamp order, with ties resolved by record identifier. Item-text records follow in the fixed candidate order. Inserted interactions are merged into the chronological history without changing the relative order of existing records. The serializer preserves this record order and the field order in Table 4; omission preserves the relative order of all retained records.

Before serialization, `&`, <, >, `"`, and `'` are encoded as `&`, `<`, `>`, `"`, and `'`, respectively. The same escaping rule applies to attribute values and field contents. Escaping operates on the raw strings, including ampersands in text that already resembles an entity. Other characters and within-field whitespace are preserved. Thus, content containing markup cannot introduce or terminate a serialized element.

Identity-based omission. For an omission set $B \subseteq \mathcal{D}_{1:T}$, the retained records at decision t are

$$\mathcal{D}_t(B) = \mathcal{D}_t \setminus (B \cap \mathcal{D}_t).$$

We construct $P_t(B)$ by replacing the `recommendation_data` block of P_t with the serialization of $\mathcal{D}_t(B)$. The user, candidate list and its order, recommendation instructions, output format, and retained field values remain unchanged, so $P_t(\varnothing) = P_t$. Selecting an identity removes its complete record at every appearance within the sequence without removing any candidate item. The same serialization rules apply to clean, observed, and omission queries in both calibration and evaluation.

B.1 Prompt Construction

Given the RRM serialization of $\mathcal{D}_t$, we construct P_t by combining the serialized records with the user identifier, fixed candidate set, and recommendation instructions. The system message specifies the ranking task and output format; the user message contains the decision-specific inputs. Candidates receive labels `C0`, `C1`, ... in their fixed input order. The returned labels are mapped back to item identities to obtain $\mathcal{R}_t$. The following template uses braced placeholders for values supplied at each decision.

System message

```
You are a recommendation ranker.  Use the user's interaction history
and item information to rank the fixed candidate items by likely
relevance.  Return only a JSON array of exactly {K} distinct candidate
label strings, most relevant first.  Do not add explanations or
recommend items outside the candidate list.
```

User message

```
User: {user_id}
Fixed candidate label-to-item-ID mapping: {candidate_mapping}
<recommendation_data>
  {serialized_records}
</recommendation_data>
Return exactly {K} different labels from {candidate_labels}. Return
labels, not numeric item IDs. Output only the JSON array.
```

Here, `candidate_mapping` is a JSON object mapping labels to item identities, and `candidate_labels` is the corresponding ordered JSON array. The two messages are formatted using the backbone's chat template. To construct the diagnostic prompt $P_t^{(B)}$, we remove the complete record elements corresponding to $B \cap \mathcal{D}_t$. The remaining records, user identifier, candidate mapping, and recommendation instructions remain unchanged.

C Case Study

This case study follows one attacked sequence through every stage of CREST. Figure 6 shows the input, RecAtom screening, calibration, final selection, and the outcome for CREST and all applicable baselines.

The sequence is a Steam deceptive-rewriting attack on Qwen2.5-7B-Instruct. The attacker targets Lugaru HD and, at the affected decisions 11–20, rewrites three item descriptions: the target and two shared distractors, Gotham pack and Inquisitor eBook. The rewrites use appealing but non-instructional text (e.g., "a rich RPG element and challenging puzzles"), while histories and candidate sets are unchanged. Without defense, Backbone returns Lugaru HD at six of the ten affected decisions, although it is never in the clean reference top-5, so its maximum item promotion $G_{\max}$ exceeds the promotion threshold η.

RecAtom screening compares the rank deterioration caused by omitting a RecAtom with its temporal influence baseline. At decision 11, omitting the interaction record Trails in the Sky and omitting the rewritten Gotham pack RecAtom both cause substantial rank deterioration. The interaction record had the same effect at decision 4, when the same user appeared, so its temporal influence baseline absorbs it, its excess influence is zero, and it is retained. The Gotham pack RecAtom has no such baseline: its influence appears only at the first affected decision, and it has the largest excess influence of all RecAtoms. CREST therefore selects it as the omission set and omits it at every decision in which it appears. This removes Lugaru HD at three decisions, even though Lugaru HD's own rewritten RecAtom remains in the prompt. The influence of an edited RecAtom therefore need not act through its own item, which is why screening measures each omission's rank deterioration across all candidate items rather than inspecting each RecAtom's own text.

Screening alone does not remove all promotion. Through its own RecAtom, Lugaru HD still appears in the screened top-5 at three decisions, which exceeds the promotion threshold. Calibration gives a calibrated limit on exclusions of three decisions per item, which exceeds the occurrence cap of two decisions set by η, so each item may appear at most twice. Final selection enforces this cap with the smallest increase in total screened rank. It replaces Lugaru HD at decision 16, where its screened rank is lowest, with the next candidate. The held-out positive at that decision is retained, and $G_{\max}$ falls to η. The baselines fail in different ways. Spotlighting and Prompt Guard 2–Record omit no manipulated RecAtom, since the rewrites contain no instructions, and Prompt Guard 2–Record returns the same recommendations as Backbone. Random-Matched omits as many RecAtoms as CREST, but its matching by record type and appearance count selects Inquisitor eBook, which has little effect on the target. RewriteDetection-Record keeps $G_{\max}$ within η, but it misses the Gotham pack RecAtom and omits nine unmodified RecAtoms. CREST keeps $G_{\max}$ within η while omitting a single manipulated RecAtom and no unmodified one.

D Solution of the Optimization Problem

D.1 Mixed-Integer Formulation of the Promotion Constraint

Let $H_\eta = \lfloor T\eta \rfloor$. For each item $i \in \mathcal{I}_{1:T}$, the promotion constraint in Equation (12) can be written as

$$a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\} \le H_\eta. \tag{i}$$

1 Input and manipulation

Steam, deceptive text rewriting; affected decisions 11–20. **Target:** Lugaru HD.
History (same user at decisions 4 and 11): Blender; Trails in the Sky; Strife.

	RecAtom	Clean excerpt		Observed excerpt
H	Trails in the Sky	interaction record		unchanged
G	Gotham pack (distractor)	"tags: Action; Free to Play"	→	"perfect for both casual and strategy gamers"
I	Inquisitor eBook (distractor)	"genres: Indie; RPG"	→	"rich sci-fi setting and engaging puzzles"
L	Lugaru HD (target)	"Martial Arts; Parkour"	→	"a rich RPG element and challenging puzzles"

↓

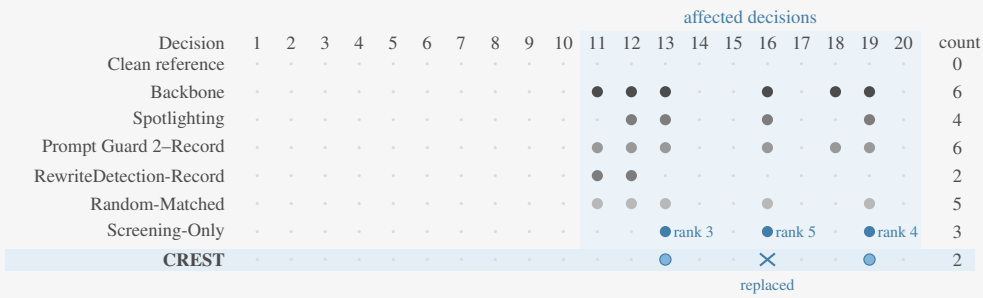
2 RecAtom screening at decision 11 (first affected decision)

Omit	Rank change after omission	Rank deterioration	Temporal influence baseline	Excess influence	
H	Inquisitor eBook: 1 → 3	0.4	0.4	0	retained
G	Lugaru HD: 2 → outside top-5	1	0	1	**omitted**
I	Lugaru HD: 2 → 3	0.25	0	0.25	retained
L	Lugaru HD: 2 → 4	0.5	0	0.5	retained

H had the same effect at decision 4, so its baseline absorbs it. Selected omission set: G, at every decision.

↓

3 Decisions at which Lugaru HD is in the returned top-5



↓

4a Calibration

Calibrated limit on exclusions: 3 decisions per item.
Occurrence cap: 2 decisions (promotion threshold 0.10).
Limit exceeds cap: each item may appear at most twice.
Lugaru HD: 3 × Inquisitor eBook: 2 ✓
All other items: at most 2 ✓

4b Final selection

Added total screened rank if Lugaru HD is replaced:
decision 13 (screened rank 3): +3
decision 16 (screened rank 5): +1 ✓
decision 19 (screened rank 4): +2
Replacement enters at rank 6; held-out positive retained.

↓

5 Outcome over the sequence (20 decisions)

Method	Omitted RecAtoms (manipulated / unmodified)	Target promotion count↓	Maximum item promotion↓
Backbone	–	6	0.300†
Spotlighting	–	4	0.200†
Prompt Guard 2–Record	0 / 0	6	0.300†
RewriteDetection-Record	2 / 9	**2**	**0.100**
Random-Matched	1 / 0	5	0.250†
Screening-Only	1 / 0	3	0.150†
CREST	1 / 0	**2**	**0.100**

† Exceeds the promotion threshold (0.10). Bold marks the best value, including ties.

Figure 6: A Steam deceptive-rewriting sequence with Qwen2.5-7B-Instruct through each stage of CREST.

Since

$$a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\} = \min\{a_i(\mathbf{z}) + b_i(\mathbf{z}), a_i(\mathbf{z}) + h_\alpha\},$$

Equation (i) is equivalent to the disjunction

$$a_i(\mathbf{z}) + b_i(\mathbf{z}) \le H_\eta \quad \text{or} \quad a_i(\mathbf{z}) + h_\alpha \le H_\eta. \tag{ii}$$

For each item $i \in \mathcal{I}_{1:T}$, introduce an auxiliary variable $y_i \in \{0, 1\}$. The disjunction in Equation (ii) has the exact mixed-integer representation

$$a_i(\mathbf{z}) + b_i(\mathbf{z}) \le H_\eta + M_{i,1} y_i, \tag{iii}$$
$$a_i(\mathbf{z}) + h_\alpha \le H_\eta + M_{i,2}(1 - y_i), \tag{iv}$$

where the nonnegative constants $M_{i,1}$ and $M_{i,2}$ make the unused inequality redundant. In particular, define

$$A_i = \left|\left\{t : i \in \mathcal{I}_t \setminus \overline{\mathcal{R}}_t\right\}\right|, \qquad B_i = \left|\left\{t : i \in \overline{\mathcal{R}}_t,\ |\mathcal{I}_t| > K\right\}\right|.$$

Since $a_i(\mathbf{z}) \le A_i$ and $b_i(\mathbf{z}) \le B_i$, valid choices are

$$M_{i,1} = \max\{A_i + B_i - H_\eta, 0\}, \qquad M_{i,2} = \max\{A_i + h_\alpha - H_\eta, 0\}.$$

When $y_i = 0$, Equation (iii) enforces the first branch of Equation (ii), while Equation (iv) is redundant. When $y_i = 1$, Equation (iv) enforces the second branch, while Equation (iii) is redundant. The formulation is therefore exact. Because $a_i(\mathbf{z})$ and $b_i(\mathbf{z})$ are linear functions of $\mathbf{z}$, replacing the promotion constraints in Equation (12) with Equations (iii)–(iv), while retaining its objective and $\mathbf{z} \in \mathcal{Z}_K$, gives a mixed-integer linear program with one auxiliary binary variable per item.

D.2 Exact Structure and Efficient Solution

Given the screened ranks, the selection problem in Equation (12) admits an exact polynomial-time solution. Let $\mathcal{T}_+ = \{t : |\mathcal{I}_t| > K\}$ denote the decisions at which alternative candidates are available. When $H_\eta \ge h_\alpha$, the screened recommendation sequence $\overline{\mathcal{R}}_{1:T}$ is feasible and optimal. When $H_\eta < h_\alpha$, the selection problem is equivalent to

$$\widehat{\mathbf{z}} \in \operatorname*{arg\,min}_{\mathbf{z} \in \mathcal{Z}_K} \sum_{t=1}^{T} \sum_{i \in \mathcal{I}_t} \bar{r}_{t,i} z_{t,i} \quad \text{s.t.} \quad \sum_{\substack{t \in \mathcal{T}_+ \\ i \in \mathcal{I}_t}} z_{t,i} \le H_\eta, \quad i \in \mathcal{I}_{1:T}, \tag{v}$$

which admits an exact minimum-cost flow formulation.

Derivation. When $H_\eta \ge h_\alpha$, the screened selection has $a_i(\mathbf{z}) = 0$ for every item. It therefore satisfies the second branch of Equation (ii). Moreover, it selects the K smallest screened ranks at each decision, minimizing the objective over $\mathcal{Z}_K$. Thus, the screened recommendation sequence is both feasible and optimal.

When $H_\eta < h_\alpha$, nonnegativity of $a_i(\mathbf{z})$ gives $a_i(\mathbf{z}) + h_\alpha > H_\eta$. The second branch of Equation (ii) is therefore impossible, and the constraint reduces to $a_i(\mathbf{z}) + b_i(\mathbf{z}) \le H_\eta$. By their definitions,

$$a_i(\mathbf{z}) + b_i(\mathbf{z}) = \sum_{\substack{t \in \mathcal{T}_+ \\ i \in \mathcal{I}_t}} z_{t,i},$$

giving Equation (v).

To solve Equation (v), construct a directed network with a source, a sink, one decision node for each $t \in \mathcal{T}_+$, and one item node for each $i \in \bigcup_{t \in \mathcal{T}_+} \mathcal{I}_t$. Connect the source to each decision node with capacity K. Connect decision t to each candidate $i \in \mathcal{I}_t$ with capacity 1 and cost $\bar{r}_{t,i}$. Finally, connect each item node to the sink with capacity H_η. All other edge costs are zero.

An integral flow of value $K|\mathcal{T}_+|$ saturates every source-to-decision edge and therefore selects exactly K candidates at each decision in $\mathcal{T}_+$. The unit-capacity decision-to-item edges give binary selections, and the item-to-sink capacities enforce Equation (v). At decisions outside $\mathcal{T}_+$, the candidate set contains exactly K items, all of which must be selected. These decisions contribute a constant to the objective and zero promotion.

Conversely, every feasible selection in Equation (v) induces an integral flow of value $K|\mathcal{T}_+|$. The flow cost equals the screened-rank cost of the selections in $\mathcal{T}_+$. Integral capacities ensure that an integral minimum-cost flow exists whenever the required flow is feasible. Such a flow therefore gives an optimal solution to Equation (12). Let $F_{\max}$ denote the maximum flow value. In this regime, an exact-K selection is feasible if and only if:

$$F_{\max} = K|\mathcal{T}_+|. \tag{vi}$$

Algorithm 1 SCREEN: Temporal Influence-Aware RecAtom Screening

Require: Recommendation inputs $(u_t, \mathcal{I}_t, \mathcal{D}_t)_{t=1}^T$; frozen recommender f_θ; $K, \beta, \lambda, W, L_{\max}$.
Set $\mathcal{D}_\cup \leftarrow \bigcup_{t=1}^T \mathcal{D}_t$.
Construct each P_t using RRM and obtain $\mathcal{R}_t = f_\theta(P_t)$ and ranks $r_{t,i}$.
Cache $\mathcal{R}_t^{(\varnothing)} \leftarrow \mathcal{R}_t$, for $t = 1, \dots, T$.
for $d \in \mathcal{D}_\cup$ **do**
    Obtain and cache $\mathcal{R}_t^{(\{d\})} = f_\theta(P_t(\{d\}))$, for $t = 1, \dots, T$.
    Compute $\delta_{t,i}(\{d\})$ using Equation (4).
end for
Compute the historical singleton baselines $\mu_{t,d,i}$ using Equation (5).
$\mathcal{B} \leftarrow \{\varnothing\}$; $\mathcal{Q} \leftarrow \{\varnothing\} \cup \{\{d\} : d \in \mathcal{D}_\cup\}$.
$\psi(\varnothing) \leftarrow 0$; $c(\varnothing) \leftarrow 0$.
for $j = 1, \dots, \min\{L_{\max}, |\mathcal{D}_\cup|\}$ **do**
    $\mathcal{E} \leftarrow \{B \cup \{d\} : B \in \mathcal{B},\ d \in \mathcal{D}_\cup \setminus B\}$.
    for $B \in \mathcal{E}$ **do**
        if $B \notin \mathcal{Q}$ **then**
            Obtain and cache $\mathcal{R}_t^{(B)} = f_\theta(P_t(B))$, for $t = 1, \dots, T$.
            Compute $\delta_{t,i}(B)$ using Equation (4).
        end if
        Compute $\mu_{t,i}(B)$ and $\psi(B)$ using Equations (3) and (6).
        $c(B) \leftarrow \frac{1}{T} \sum_{t:\, |\mathcal{D}_t|>0} \frac{|B \cap \mathcal{D}_t|}{|\mathcal{D}_t|}$.
    end for
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \mathcal{E}$.
    Retain in $\mathcal{B}$ the $\min\{W, |\mathcal{E}|\}$ subsets of $\mathcal{E}$ with largest $\psi(B) - \lambda c(B)$, using the fixed tie-breaking rule.
end for
$\widehat{B} \leftarrow \arg\max_{B \in \mathcal{Q}} \{\psi(B) - \lambda c(B)\}$, using the same tie-breaking rule.
return $\overline{\mathcal{R}}_{1:T} = (\mathcal{R}_t^{(\widehat{B})})_{t=1}^T$.

Algorithm 2 CREST

Require: N calibration sequences $(u_t^{(\ell)}, \mathcal{I}_t^{(\ell)}, \mathcal{D}_t^{(\ell)})_{t=1}^T$ and their reference recommendations $\mathcal{R}_{1:T}^{*,(\ell)}$; evaluated inputs $(u_t, \mathcal{I}_t, \mathcal{D}_t)_{t=1}^T$; $f_\theta, K, \alpha, \eta, \beta, \lambda, W, L_{\max}$.
for $\ell = 1, \dots, N$ **do**
    $\overline{\mathcal{R}}_{1:T}^{(\ell)} \leftarrow$ SCREEN$(\ (u_t^{(\ell)}, \mathcal{I}_t^{(\ell)}, \mathcal{D}_t^{(\ell)})_{t=1}^T; f_\theta, K, \beta, \lambda, W, L_{\max})$.
    $S_\ell \leftarrow \max_{i \in \bigcup_{t=1}^T \mathcal{I}_t^{(\ell)}} G_i\Big(\overline{\mathcal{R}}_{1:T}^{(\ell)}, \mathcal{R}_{1:T}^{*,(\ell)}\Big)$.
end for
$\widehat{q}_\alpha \leftarrow [S_1, \dots, S_N, 1]_{(\lceil (N+1)(1-\alpha) \rceil)}$.
$\overline{\mathcal{R}}_{1:T} \leftarrow$ SCREEN$(\ (u_t, \mathcal{I}_t, \mathcal{D}_t)_{t=1}^T; f_\theta, K, \beta, \lambda, W, L_{\max})$.
Assign each candidate its screened rank $\bar{r}_{t,i}$, using $K+1$ for items outside $\overline{\mathcal{R}}_t$.
$h_\alpha \leftarrow \lfloor T \widehat{q}_\alpha \rfloor$.
Solve Equation (12) using the formulation in Appendix D.
if the optimization is infeasible **then**
    return `infeasible`.
end if
Let $\widehat{\mathbf{z}}$ be the resulting optimal selection.
$\widehat{\mathcal{R}}_t \leftarrow \{i \in \mathcal{I}_t : \widehat{z}_{t,i} = 1\}$, for $t = 1, \dots, T$.
return $\widehat{\mathcal{R}}_{1:T}$.

The network has $O\Big(\sum_{t \in \mathcal{T}_+} |\mathcal{I}_t|\Big)$ edges. Thus, given the screened ranks, feasibility and optimal selection can be determined in polynomial time using maximum-flow and minimum-cost flow algorithms, respectively (Ahuja et al., 1995).

E ALGORITHMS

Algorithm 1 details the RecAtom screening procedure, and Algorithm 2 combines screening, calibration, and final recommendation selection. We use beam width $W \geq 1$ and maximum omission depth $L_{\max} \geq 1$, with all screening settings fixed before calibration. Diagnostic rankings are cached within each sequence; when an omission leaves a prompt unchanged, its original ranking is reused. The calibration threshold is computed once and reused for subsequent evaluated sequences.

F PROOFS

F.1 PROOF OF PROPOSITION 1

Proof. We begin by indexing the N calibration sequence pairs by $\ell = 1, \dots, N$ and the evaluated sequence pair by $\ell = N + 1$. For each $\ell \in \{1, \dots, N + 1\}$, define

$$S_\ell = \max_{i \in \mathcal{I}_{1:T}^{(\ell)}} G_i\left(\overline{\mathcal{R}}_{1:T}^{(\ell)}, \mathcal{R}_{1:T}^{*,(\ell)}\right). \tag{i}$$

By assumption, the scores $S_1, \dots, S_{N+1}$ are exchangeable.

Let $k_\alpha = \lceil (N + 1)(1 - \alpha) \rceil$. Recall that $\widehat{q}_\alpha$ is the k_α-th ordered value of $S_1, \dots, S_N, 1$. The scores satisfy $S_\ell \in [0, 1]$. Let R_{N+1} denote the rank of S_{N+1} among $S_1, \dots, S_{N+1}$, with ties broken uniformly at random. By exchangeability, R_{N+1} is uniform over $\{1, \dots, N + 1\}$, giving

$$\Pr[R_{N+1} > k_\alpha] = \frac{N + 1 - k_\alpha}{N + 1}. \tag{ii}$$

Whenever $S_{N+1} > \widehat{q}_\alpha$, we have $\widehat{q}_\alpha < 1$. Thus, at least k_α calibration scores are at most $\widehat{q}_\alpha$ and hence strictly smaller than S_{N+1}, which implies $R_{N+1} > k_\alpha$. Using Eq. (ii), we obtain

$$\Pr[S_{N+1} > \widehat{q}_\alpha] \le \Pr[R_{N+1} > k_\alpha] = \frac{N + 1 - k_\alpha}{N + 1} \le \alpha. \tag{iii}$$

Taking the complement of Eq. (iii) and identifying the pair indexed by $N + 1$ with the evaluated sequence gives

$$\Pr\left[\max_{i \in \mathcal{I}_{1:T}} G_i\left(\overline{\mathcal{R}}_{1:T}, \mathcal{R}_{1:T}^*\right) \le \widehat{q}_\alpha\right] \ge 1 - \alpha. \tag{iv}$$

Finally, since $\mathcal{R}_t^* \in \binom{\mathcal{I}_t}{K}$ for every t, Equation (9) gives

$$\max_{i \in \mathcal{I}_{1:T}} G_i\left(\overline{\mathcal{R}}_{1:T}, \mathcal{R}_{1:T}^*\right) \le \widehat{q}_\alpha \quad \Longleftrightarrow \quad \mathcal{R}_{1:T}^* \in \mathfrak{C}_\alpha\left(\overline{\mathcal{R}}_{1:T}\right). \tag{v}$$

Combining Eqs. (iv) and (v) proves the proposition. □

F.2 PROOF OF LEMMA 1

Proof. Fix $\mathbf{z} \in \mathcal{Z}_K$ and $i \in \mathcal{I}_{1:T}$, and consider any $\mathcal{C}_{1:T} \in \mathfrak{C}_\alpha(\overline{\mathcal{R}}_{1:T})$. The selected decisions at which $i \notin \overline{\mathcal{R}}_t$ contribute at most $a_i(\mathbf{z})$. When $i \in \overline{\mathcal{R}}_t$ and $|\mathcal{I}_t| = K$, the exact-K condition forces $\mathcal{C}_t = \mathcal{I}_t = \overline{\mathcal{R}}_t$, so i cannot be excluded. All remaining contributions must therefore come from the decisions counted by $b_i(\mathbf{z})$.

Membership in the calibrated family gives

$$\sum_{t:\, i \in \overline{\mathcal{R}}_t} \mathbb{I}[i \notin \mathcal{C}_t] = T\, G_i\left(\overline{\mathcal{R}}_{1:T}, \mathcal{C}_{1:T}\right) \le T\widehat{q}_\alpha. \tag{i}$$

Since the left-hand side of Equation (i) is integer-valued,

$$\sum_{t:\, i \in \overline{\mathcal{R}}_t} \mathbb{I}[i \notin \mathcal{C}_t] \le \lfloor T\widehat{q}_\alpha \rfloor = h_\alpha. \tag{ii}$$

Combining the two types of selected decisions yields

$$\sum_{t:\, i \in \mathcal{I}_t} z_{t,i} \mathbb{I}[i \notin \mathcal{C}_t] \le a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\}. \tag{iii}$$

Taking the maximum over the calibrated family in Equation (iii) gives

$$\widehat{G}_i(\mathbf{z}) \le \frac{a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\}}{T}. \tag{iv}$$

It remains to show that this bound is attained. Let $\mathcal{J}$ be any set of $\min\{b_i(\mathbf{z}), h_\alpha\}$ decisions counted by $b_i(\mathbf{z})$. For each $t \in \mathcal{J}$, the condition $|\mathcal{I}_t| > K$ and the fact that $|\overline{\mathcal{R}}_t| = K$ ensure the existence of an item $j_t \in \mathcal{I}_t \setminus \overline{\mathcal{R}}_t$. Define

$$\mathcal{C}_t^\star = \begin{cases} \left(\overline{\mathcal{R}}_t \setminus \{i\}\right) \cup \{j_t\}, & t \in \mathcal{J}, \\ \overline{\mathcal{R}}_t, & t \notin \mathcal{J}. \end{cases} \tag{v}$$

Every $\mathcal{C}_t^\star$ contains exactly K items. Moreover, the construction removes only i from the screened recommendations, and does so at $|\mathcal{J}|$ decisions. Hence,

$$G_i\left(\overline{\mathcal{R}}_{1:T}, \mathcal{C}_{1:T}^\star\right) = \frac{|\mathcal{J}|}{T} \le \widehat{q}_\alpha, \qquad G_j\left(\overline{\mathcal{R}}_{1:T}, \mathcal{C}_{1:T}^\star\right) = 0 \quad \text{for every } j \ne i. \tag{vi}$$

Table 5: Dataset statistics after preprocessing. Users have retained interactions; items count the metadata catalog; review texts count nonempty bodies among retained interactions. RecAtom counts are averaged over ten clean test sequences per dataset ($T = 20$); sequence counts use distinct identities.

Dataset	Users	Items	Interactions	Review texts	Atoms/decision	Atoms/sequence
Amazon Beauty	319,319	32,487	366,207	365,812	53	984.6
Steam	2,567,538	32,131	7,793,069	7,775,106	53	987.7
MovieLens-1M	6,040	3,883	1,000,209	0	53	902.5

Therefore, $\mathcal{C}^\star_{1:T} \in \mathfrak{C}_\alpha(\overline{\mathcal{R}}_{1:T})$.

At every decision counted by $a_i(\mathbf{z})$, Equation (v) sets $\mathcal{C}^\star_t = \overline{\mathcal{R}}_t$, which already excludes i. The construction also excludes i at every decision in $\mathcal{J}$. Consequently,

$$\frac{1}{T}\sum_{t:\, i\in\mathcal{I}_t} z_{t,i}\mathbb{I}[i \notin \mathcal{C}^\star_t] = \frac{a_i(\mathbf{z}) + |\mathcal{J}|}{T} = \frac{a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\}}{T}. \tag{vii}$$

Equation (vii) attains the upper bound in Equation (iv), proving Equation (11). □

F.3 Proof of Theorem 1

Proof. Let $\widehat{\mathbf{z}}$ be any feasible solution of Equation (12), which exists almost surely by assumption. Since $\widehat{\mathbf{z}} \in \mathcal{Z}_K$, the resulting sets $\widehat{\mathcal{R}}_t = \{i \in \mathcal{I}_t : \widehat{z}_{t,i} = 1\}$ contain exactly K items at every decision. Using Lemma 1, the promotion constraints give, for every $i \in \mathcal{I}_{1:T}$:

$$\widehat{G}_i(\widehat{\mathbf{z}}) = \frac{a_i(\widehat{\mathbf{z}}) + \min\{b_i(\widehat{\mathbf{z}}), h_\alpha\}}{T} \le \frac{\lfloor T\eta \rfloor}{T} \le \eta. \tag{i}$$

Let $\mathcal{E} = \{\mathcal{R}^*_{1:T} \in \mathfrak{C}_\alpha(\overline{\mathcal{R}}_{1:T})\}$ denote the coverage event. By Proposition 1, we have:

$$\Pr(\mathcal{E}) \ge 1 - \alpha. \tag{ii}$$

On $\mathcal{E}$, the reference sequence $\mathcal{R}^*_{1:T}$ belongs to the family over which $\widehat{G}_i$ is maximized. Thus, Equation (10) and Lemma 1 give the following bound simultaneously for every $\mathbf{z} \in \mathcal{Z}_K$ and $i \in \mathcal{I}_{1:T}$:

$$\frac{1}{T}\sum_{t:\, i\in\mathcal{I}_t} z_{t,i}\mathbb{I}[i \notin \mathcal{R}^*_t] \le \widehat{G}_i(\mathbf{z}) = \frac{a_i(\mathbf{z}) + \min\{b_i(\mathbf{z}), h_\alpha\}}{T}. \tag{iii}$$

Taking $\mathbf{z} = \widehat{\mathbf{z}}$ in Equation (iii) and applying Equation (i), we obtain:

$$\begin{aligned} G_i\Big(\widehat{\mathcal{R}}_{1:T}, \mathcal{R}^*_{1:T}\Big) &= \frac{1}{T}\sum_{t:\, i\in\mathcal{I}_t} \widehat{z}_{t,i}\mathbb{I}[i \notin \mathcal{R}^*_t] \\ &\le \widehat{G}_i(\widehat{\mathbf{z}}) \le \eta, \qquad i \in \mathcal{I}_{1:T}. \end{aligned} \tag{iv}$$

Equation (iv) holds for every item on the same event $\mathcal{E}$ whenever the selection is feasible. Since feasibility holds almost surely, Equation (ii) yields:

$$\Pr\left[\max_{i\in\mathcal{I}_{1:T}} G_i\Big(\widehat{\mathcal{R}}_{1:T}, \mathcal{R}^*_{1:T}\Big) \le \eta\right] \ge \Pr(\mathcal{E}) \ge 1 - \alpha. \tag{v}$$

This proves the theorem. □

G Experimental Details

This appendix describes dataset preparation, attack construction, baseline adaptations, and the evaluation protocol.

G.1 Datasets, Preprocessing, and Sequence Construction

We evaluate CREST on Amazon Beauty (the All_Beauty category of Amazon Reviews 2018), Steam Video Game Data version 2, and MovieLens-1M. Beauty and Steam provide review interactions and item metadata, whereas MovieLens provides ratings, titles, and genres. Text manipulations use candidate item metadata; for MovieLens, we use its native titles and genres without external text enrichment. Table 5 summarizes the datasets after preprocessing.

Preprocessing. We retain interactions with valid identifiers, timestamps, and corresponding item metadata containing a title, without k-core filtering. Ratings of at least four define positive interactions for Beauty and MovieLens; Steam reviews are treated as implicit positive interactions regardless of sentiment. Lower-rated interactions remain in user histories, and interactions with missing review text are retained. Distinct interaction rows receive separate RecAtom identities; duplicate item metadata

uses the first valid entry. Titles and descriptions are truncated to 240 characters per field, and candidate descriptions must contain at least two whitespace-separated tokens.

Histories and candidates. For each recommendation decision, we hold out a positive interaction and construct the history from the user's three most recent interactions strictly before its timestamp. Among positive interactions sharing a user and timestamp, we select the one with the lowest source event identifier. Each clean prompt contains three history RecAtoms and 50 candidate item-text RecAtoms, giving 53 records.

We sample one promotion target per sequence from the bottom 90% of eligible items ranked by training interaction count, excluding items with no training interactions and all held-out positives in the sequence. Each candidate set contains the positive item, the target, two distractors shared across the sequence, and 46 additional items sampled uniformly without replacement. Shared distractors exclude all sequence positives; the remaining distractors may include previously interacted items. Candidate sets and their order remain fixed across corresponding clean, observed, diagnostic, and defended queries. With one designated positive per decision, Recall@K equals hit rate in this sampled-candidate setting.

Sequences and data splits. We partition interactions chronologically into training, development, calibration, and test intervals using the 70th, 80th, and 90th percentile timestamps; boundary ties enter the later interval. Each sequence samples distinct user–cutoff windows, includes two decisions from the same user, and orders decisions chronologically. Sequences may span multiple users. History RecAtoms recur only in subsequent windows for the same user, whereas item-text RecAtoms recur when an item reappears as a candidate.

For Beauty and Steam, a fixed projection reduces the original 100-decision sequences to $T = 20$, retaining ten decisions before the attack window and ten within it, including the repeated-user pair in the latter half. MovieLens sequences are constructed directly at $T = 20$. The main comparisons use 50 calibration sequences and ten test sequences across five seed groups per dataset–attack setting. For RQ3, we instead use 20 independent blocks, each containing 19 calibration and five test sequences drawn from the same fixed held-out sequence generator.

Clean counterparts preserve the original contents of edited records and exclude inserted records. Clean test outputs and manipulation labels are used only for evaluation; CREST receives clean–observed pairs for calibration and observed inputs at test time.

G.2 Attack Construction and Attacker Capabilities

We consider instruction injection, deceptive text rewriting, and interaction-history manipulation, each aiming to promote one target item across a recommendation sequence. The attacker may modify permitted content fields or insert interaction records through trusted ingestion, which assigns record identities and references to existing catalog items. Existing identities, references, RecAtom boundaries, candidate sets, and recommendation instructions remain fixed. Targets are selected before attack generation following Appendix G.1, and evaluation does not condition on attack success. The default setting applies manipulation to the final ten decisions of a 20-decision sequence. Instruction injection and history manipulation require no ranking feedback, whereas deceptive rewriting uses observed rankings to refine the manipulated text.

Instruction injection. We append an instruction to rank the target first to the description fields of three item-text RecAtoms per affected decision: the target record and two additional candidate records sampled using a fixed seed. The three-record budget follows from rounding the nominal 5% manipulation intensity upward over 53 records. Each selected identity is assigned one of two prespecified direct templates and retains the same payload whenever it is manipulated again. RRM serialization escapes reserved markup so that the payload cannot alter record boundaries. For the recurrence analysis in Section 5.2.2, we hold three record identities and their payloads fixed and vary the decisions in which they appear.

Deceptive text rewriting. We adapt TextSimu's popularity-guided rewriting to manipulate the apparent relevance conveyed by candidate item descriptions. The target description and two shared distractor descriptions are rewritten independently toward the same promotion objective, then combined in the observed sequence. Qwen2.5-7B-Instruct generates the rewritten text and provides embeddings for reference retrieval. The reference pool contains the 1,000 eligible items with the most training interactions. For each description, we retrieve 50 references by cosine similarity, select 20 keywords,

and use five personas to draft and refine the text. Refinement permits up to two discussion rounds with three revision attempts per round. Each generation is limited to 768 new tokens, with at most 200 ranking-feedback queries per rewritten record.

A constrained prefix preserves the source title and numeric expressions. For Beauty, the retrieved reference set is adjusted when necessary to provide 40 distinct candidates for keyword selection. Feedback measures the target's top-10 recommendation frequency. Beauty and Steam attacks are generated on the original 100-decision sequences and evaluated using top-5 recommendations on the fixed 20-decision projection described in Appendix G.1. The additional backbone comparisons reuse these generated attacks.

Interaction-history manipulation. Following the inference-time history-insertion setting studied by RETURN, we use a prespecified rule to insert three positive interaction records per affected user: one for the target and two for items from the user's recent history. These filler items are drawn from the most recent distinct historical items, excluding the target, with repetition when fewer than two are available. Original history records and held-out positives remain unchanged. Inserted timestamps fall after the latest original history event and any earlier recommendation decision for that user, but before the first affected decision. The inserted RecAtoms retain their identities and recur in subsequent history windows for the same user.

G.3 Baseline Implementations and Information Access

Within each setting, all methods use the same frozen recommender, observed sequences, candidate sets, and decoding protocol. Baseline thresholds and implementation settings are prespecified and fixed throughout evaluation. Test clean outputs and manipulation labels are used only to compute evaluation metrics; CREST receives clean–observed pairs during calibration. Record omissions apply to every appearance of the selected identity while preserving all candidate items in $\mathcal{I}_t$.

- **Backbone** directly queries the frozen recommender on the observed recommendation data. The main experiments use Qwen2.5-7B-Instruct (Yang et al., 2024); additional backbone settings are reported in Appendix H.1. No records are removed and no final promotion constraint is applied.
- **Spotlighting** (Hines et al., 2024) uses datamarking to distinguish record contents from recommendation instructions. We insert a caret marker between whitespace-separated words in each record field and add an instruction to treat the marked text as data. Record identities, structural boundaries, and candidate sets are retained.
- **Prompt Guard 2–Record** (Chennabasappa et al., 2025) applies the frozen Llama-Prompt-Guard-2-86M detector to each RecAtom's field contents. Long inputs are divided into chunks of at most 510 content tokens, reserving two positions for special tokens, with a 128-token overlap. The record score is the maximum malicious-class probability across chunks and appearances; identities scoring at least 0.5 are omitted.
- **RewriteDetection-Record** (Wang et al., 2025) adapts RewriteDetection to record removal under deceptive text rewriting. Each description is split at its midpoint between whitespace-separated tokens, and Qwen2.5-7B-Instruct generates three continuations from the prefix. Generation uses temperature 0.7, top-p $= 0.9$, and a limit of 128 new tokens. The detection score combines unigram and bigram continuation overlap with changes in top-5 recommendation frequency, using equal component weights and a reference cohort drawn from training data. An identity is omitted when its maximum score across appearances exceeds 0.5. Its associated item remains eligible for recommendation.
- **RETURN-Del** (Ning et al., 2026) adapts RETURN's collaborative scoring to interaction-history manipulation. We construct a sparse item co-occurrence graph from training histories, retaining position gaps of up to five interactions, and apply the original row-normalized support calculation. Scores are aggregated by taking the minimum across appearances of each interaction identity; identities with support below 10^{-12} are omitted. Items absent from the graph receive zero support. This adaptation uses deletion without history replacement or ensemble voting.
- **Random-Matched** is a randomized control that matches CREST's realized screening omissions without using manipulation labels. We group record identities by record type, appearance count, and the multiset of prompt record counts at their appearances, then sample without replacement the same number of omissions as CREST within each group. This preserves the number of distinct omitted records and the omission cost $c(\widehat{B})$ exactly. We use one draw with the sequence seed plus 71 and allow overlap with CREST's omissions. The recommender is queried after these omissions, without final selection. Comparison with Screening-Only assesses selective screening under matched omission cost.

Table 6: Experimental settings. Main-comparison defaults are retained unless varied in the indicated analysis. RQ3 uses an independent calibration-block design with 100 test sequences in total.

Quantity	Default	Variation or evaluation scope
List size K	5	Fixed
Candidate size $\lvert\mathcal{I}_t\rvert$	50	Fixed
Sequence length T	20	Fixed
Calibration pairs N	50	19 per block in RQ3
Calibration tolerance α	0.10	0.01, 0.025, 0.05, 0.10, 0.20 (RQ3)
Promotion threshold η	0.10	0.02, 0.05, 0.10, 0.15, 0.20 (RQ3)
Temporal decay β	0.05	Fixed
Penalty coefficient λ	0.05	Fixed
Beam width W	1	Fixed
Maximum omission depth $L_{\max}$	1	Fixed
Manipulation intensity (%)	5	1, 2.5, 5, 10, 20 (RQ2)
Recurrence (%)	50	10, 25, 50, 75, 100 (RQ2)
Evaluation seed groups	5	Seeds 11, 23, 37, 53, 71; two test sequences per group
RQ3 calibration blocks B_{cal}	20	Independent calibration draws
RQ3 test sequences per block L	5	One calibration threshold shared within each block

G.4 Implementation Details

We use Qwen2.5-7B-Instruct (Yang et al., 2024) for the main experiments and Llama-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen3.5-9B (Qwen Team, 2026) for the additional backbone comparisons. All ranking models remain frozen and use BF16 precision, greedy decoding, and their checkpoint-specific tokenizers and chat templates. Qwen3.5 is used in text-only mode with thinking disabled. Screening parameters are prespecified and fixed before calibration. Table 6 summarizes the experimental settings and the parameter variations examined in RQ2 and RQ3.

Inference and serialization. Each ranking call processes one prompt, with a maximum of 12,288 input tokens and 256 generated tokens. Constrained decoding produces a JSON array of exactly K distinct candidate labels, which is validated without output repair. RRM serialization preserves RecAtom identities, field order, and reserved-character escaping. Dataset construction uses 240-character title and description prefixes; each resulting RecAtom remains a complete omission unit. Inputs exceeding the token budget raise an error without further record truncation. Exact-query caches are keyed by the serialized prompt, ordered candidates, K, decoding configuration, and model/runtime identity.

Screening and calibration. With $L_{\max}$ choosen, screening evaluates the empty omission set and every single-identity omission. An omitted identity is removed at all its appearances. Temporal reference values use preceding appearances within the same sequence and reset at sequence boundaries. Calibration uses the $\lceil (N+1)(1-\alpha) \rceil$-th order statistic after appending the endpoint 1. Each backbone and the Without Screening ablation receive separate calibration. The RQ3 parameter comparisons reuse fixed screened rankings and recompute calibration thresholds and final selection for each setting.

Final selection. Let $H_\eta = \lfloor \eta T \rfloor$. Following Appendix D.2, CREST returns $\overline{\mathcal{R}}_{1:T}$ when $H_\eta \geq h_\alpha$. Otherwise, a capacitated min-cost flow problem enforces the occurrence cap and minimizes screened-rank cost. The cap applies to decisions with more than K candidates; decisions with exactly K candidates have fixed outputs. Items outside the screened top-K list receive rank $K+1$. Selected items are ordered by screened rank, with ties broken by the fixed candidate order. Infeasible sequences receive no exact-K output; inference and execution failures are recorded separately.

Computing environment. Experiments run on NVIDIA H100 NVL and RTX A5000 GPUs. Qwen2.5 and Llama-3.1 use PyTorch 2.4.1 with CUDA 12.1, Transformers 4.44.2, and tokenizers 0.19.1. Qwen3.5 uses PyTorch 2.6.0 with CUDA 12.4, Transformers 5.3.0, and tokenizers 0.22.2, with the native PyTorch linear-attention implementation. Final selection runs on the CPU using NetworkX's min-cost flow solver. The controlled inference-timing comparison in RQ4 uses H100 NVL GPUs.

G.5 Evaluation Metrics

Recommendation quality and promotion. Each decision has one held-out relevant item $y_t \in \mathcal{I}_t$. Let $s_{t,j}$ denote the item at rank j in the returned list $\mathcal{S}_t$. We compute

$$\mathrm{Recall@}K(t) = \mathbb{I}[y_t \in \mathcal{S}_t],$$

$$\mathrm{NDCG@}K(t) = \sum_{j=1}^{K} \frac{\mathbb{I}[s_{t,j} = y_t]}{\log_2(j+1)}.$$

Both metrics are averaged over decisions within each sequence and reported as percentages, with $K = 5$ in the main evaluation. Promotion is measured by $G_{\max}$, defined in Equation (14), using the clean recommendations as the reference and the sequence length T as the denominator. We report the mean of the sequence-level $G_{\max}$ values on the $[0, 1]$ scale.

False-positive rate. Let $\mathcal{A}$ contain the distinct RecAtom identities manipulated at least once within a sequence, and let $\mathcal{D}_{1:T} = \bigcup_{t=1}^{T} \mathcal{D}_t$ contain all observed identities. For omission set $\widehat{B}$, the false-positive rate is

$$\mathrm{FPR} = \frac{|\widehat{B} \setminus \mathcal{A}|}{|\mathcal{D}_{1:T} \setminus \mathcal{A}|}.$$

FPR measures the fraction of unmodified RecAtoms removed, counting each identity once per sequence regardless of its recurrence. We report FPR as a percentage for methods that select records for omission; it is not applicable to Backbone or Spotlighting.

Aggregation and uncertainty. Comparisons use the same sequences, candidate sets, targets, and attack payloads. Recommendation quality and promotion are compared on the sequences returned by every method in the comparison. The main comparisons average ten sequences across five seed groups, with two sequences per group. FPR is computed separately for each sequence and then averaged. Statistical-control experiments use 20 independent calibration blocks, each containing five test sequences. Their 95% confidence intervals resample whole calibration blocks, keeping sequences that share a calibration threshold together.

H Additional Experimental Results

H.1 Additional Results for Overall Effectiveness (RQ1)

We repeat instruction injection on Amazon Beauty, deceptive rewriting on Steam, and history manipulation on MovieLens-1M with Llama-3.1-8B-Instruct and Qwen3.5-9B. We retain the same attacks and recalibrate CREST for each backbone. Qwen3.5-9B uses non-thinking inference.

As shown in Table 7, CREST achieves the best or tied-best Recall@1, Recall@5, and NDCG@5, together with the lowest mean $G_{\max}$, in all three settings for both backbones. Its mean $G_{\max}$ ranges from 0.040 to 0.095, remaining below $\eta = 0.10$ throughout. Under rewriting and history manipulation, it also achieves lower FPR than the corresponding attack-specific baselines. These results support CREST's effectiveness across LLM backbones and recommendation domains.

H.2 Additional Results on Attack Timing (RQ2)

We fix the same three corrupted records and ten affected decisions within each twenty-decision sequence. Manipulation appears in the first ten decisions (early burst), the last ten (delayed onset), or a dispersed schedule. Calibration remains unchanged. The dispersed condition is the 50% recurrence reference used in Figure 4.

Table 8 shows that CREST achieves the highest NDCG@5 and lowest mean $G_{\max}$ under all three attack timings. Delayed onset produces the highest promotion for Backbone and Prompt Guard 2–Record, whereas dispersed manipulation leaves CREST with its highest residual promotion. CREST nevertheless remains below $\eta = 0.10$ throughout. All three methods obtain their highest NDCG@5 under an early burst and their lowest under delayed onset, showing that the same attack volume can have different effects depending on when it appears. CREST's advantage across these schedules supports robustness beyond a particular attack timing.

H.3 Hyperparameter Analysis

We consider the impact of temporal decay β, penalty coefficient λ, and beam width W on recommendation quality and promotion control using Amazon Beauty (Table 9) with Qwen2.5-7B-Instruct backbone model.

Table 7: Effectiveness with Llama-3.1-8B-Instruct and Qwen3.5-9B on three dataset–attack settings, averaged over five seeds. R (Recall), N (NDCG), and FPR are in %; $G_{\max}$ is computed at top-5 and averaged across sequences. All methods return all ten sequences. Bold/underline indicate best/second-best values per backbone and setting. † marks mean $G_{\max} > \eta = 0.10$.

	Llama-3.1-8B-Instruct					Qwen3.5-9B				
Method	R@1↑	R@5↑	N@5↑	$G_{\max}$ ↓	FPR↓	R@1↑	R@5↑	N@5↑	$G_{\max}$ ↓	FPR↓
Amazon Beauty: instruction injection										
Backbone	2.50	16.50	9.07	0.335†	–	3.50	11.50	7.80	0.445†	–
Spotlighting	2.50	17.50	9.93	0.210†	–	**5.00**	15.00	**10.33**	0.250†	–
Prompt Guard 2–Record	3.00	17.50	10.02	0.340†	0.075	3.50	15.00	10.09	0.400†	0.097
Random-Matched	2.50	15.50	8.64	0.335†	0.069	4.50	12.00	7.98	0.440†	**0.081**
CREST	**4.50**	**18.50**	**11.73**	**0.070**	**0.063**	**5.00**	**16.50**	**10.33**	**0.095**	0.083
Steam: deceptive text rewriting										
Backbone	1.50	11.50	7.81	0.080	–	13.00	25.00	20.50	0.080	–
Spotlighting	1.00	10.50	6.35	0.060	–	11.50	25.00	18.21	0.105†	–
Prompt Guard 2–Record	3.50	13.00	8.49	0.080	0.095	15.00	29.00	21.84	0.080	0.081
RewriteDetection-Record	3.50	13.00	8.54	0.050	1.310	14.50	30.00	22.25	0.085	1.320
Random-Matched	3.50	12.50	8.29	0.045	0.091	15.00	30.00	22.65	0.075	**0.031**
CREST	**4.00**	**13.50**	**9.09**	**0.040**	**0.071**	**15.50**	**30.50**	**22.91**	**0.070**	**0.031**
MovieLens-1M: history manipulation										
Backbone	3.50	14.50	9.26	0.135†	–	3.50	16.00	10.08	0.390†	–
Spotlighting	3.50	14.00	8.94	0.135†	–	4.50	**17.50**	11.19	0.430†	–
Prompt Guard 2–Record	3.50	14.50	9.26	0.135†	0.033	3.50	16.00	10.08	0.390†	0.044
RETURN-Del	5.00	**15.00**	9.78	0.075	1.740	**6.50**	16.00	12.03	0.120†	1.740
Random-Matched	3.50	14.50	9.37	0.145†	0.078	3.00	16.00	9.99	0.380†	0.078
CREST	**6.50**	**15.00**	**10.08**	**0.060**	**0.027**	**6.50**	**17.50**	**12.18**	**0.085**	**0.035**

Table 8: Recommendation performance and promotion control under different attack timings on Amazon Beauty with Qwen2.5-7B-Instruct. R, N, and FPR denote Recall@5, NDCG@5, and false-positive rate, respectively, all in percent; $G_{\max}$ is averaged across sequences. Results average ten sequences across five seeds. Bold and underlining indicate the best and second-best results, including displayed ties; † marks mean $G_{\max} > 0.10$.

Method	R↑	N↑	$G_{\max}$ ↓	FPR↓
Early burst				
Backbone	12.00	7.25	0.110†	–
Prompt Guard 2–Record	11.50	7.43	0.115†	0.052
CREST	**12.50**	**8.08**	**0.075**	**0.041**
Delayed onset				
Backbone	10.00	5.21	0.140†	–
Prompt Guard 2–Record	10.50	5.30	0.145†	0.059
CREST	**11.00**	**6.03**	**0.085**	**0.041**
Dispersed				
Backbone	10.00	6.17	0.120†	–
Prompt Guard 2–Record	**10.50**	6.25	0.135†	0.063
CREST	**10.50**	**6.44**	**0.090**	**0.031**

Temporal decay. Increasing β places greater weight on recent omission responses, allowing the historical reference to adapt more quickly. The values show modest utility changes, with promotion remaining between 0.085 and 0.090 across decay rates. Such a pattern suggests that emphasizing recent evidence may help recommendation quality without necessarily impacting promotion control.

Penalty coefficient. The λ comparison represents a possible balance between suppressing manipulated evidence and preserving useful context. With $\lambda = 0$, screening may discard relevant information alongside influential records. An intermediate penalty can discourage unnecessary removal, whereas a large penalty may retain manipulated evidence. The values therefore depicts improved utility at an intermediate penalty, followed by weaker utility and promotion control when removal is strongly discouraged.

Beam width and computational cost. At $L_{\max} = 2$, increasing W retains more single-identity candidates for expansion into joint omissions. The values depict gains from a wider search followed by a plateau.

H.4 Time and Space Complexity

In this section, we analyze the time and space complexity of CREST. Calibration is performed offline for a fixed model and screening configuration. Given N calibration scores, threshold computation

Table 9: Hyperparameter analysis on Amazon Beauty dataset under instruction injection with Qwen2.5-7B-Instruct.

Parameter	Value	R↑	N↑	$G_{\max}$ ↓
β	0.00	13.00	7.28	0.090
	0.01	13.50	7.43	0.085
	0.05	13.50	7.38	0.090
	0.10	13.50	7.55	0.090
λ	0.00	13.00	7.31	0.080
	0.05	13.50	7.38	0.090
	0.10	14.00	7.62	0.085
	0.20	13.50	7.39	0.090
	0.50	12.50	7.12	0.095
W	1	13.50	7.38	0.090
	2	13.50	7.61	0.080
	4	14.00	7.73	0.080
	8	14.00	7.73	0.080

requires $\mathcal{O}(N \log N)$ time and $\mathcal{O}(N)$ temporary space. The resulting threshold occupies $\mathcal{O}(1)$ storage and is reused across recommendation sequences. This computation depends on the calibration size.

During inference, screening operates on the M distinct RecAtoms present in the current sequence. Consequently, its workload is governed by the sequence length and recommendation-context budget. With the reported $L_{\max} = 1$, screening evaluates the empty omission set and all singleton omissions. Exact-query caching reuses unchanged prompts, limiting fresh ranking generations to at most $T + \sum_{t=1}^{T} |\mathcal{D}_t|$. The stored rankings and influence statistics require $\mathcal{O}(MTK)$ auxiliary space, excluding model weights and inference caches.

Final selection then applies the calibrated constraint to the screened rankings.

The measurements in Table 3 show that final selection requires only 18–19 ms per 20-decision sequence. These results demonstrate that the selection stage enforces calibrated promotion constraints with low additional latency, resulting in final-stage recommendation time comparable to Backbone.