Learning Evidence Highlighting for Frozen LLMs

Shaoang Li[1,*] **Yanhang Shi**[1,*] **Yufei Li**[2] **Mingfu Liang**[2] **Xiaohan Wei**[2]
Yunchen Pu[2] **Fei Tian**[2] **Chonglin Sun**[2] **Frank Shyu**[2] **Sandeep Pandey**[2]
Luke Simon[2] **Xi Liu**[2,†] **Jian Li**[1,†]

[1]Stony Brook University [2]Meta AI

Abstract

Large Language Models (LLMs) can reason well over focused inputs, yet often miss decisive evidence buried in long, noisy contexts. We introduce HILIGHT, an Evidence Emphasis framework that turns evidence selection into a learned, non-destructive input-side control problem for frozen LLMs. Rather than retrieving, pruning, compressing, or rewriting the input, HILIGHT trains a lightweight Emphasis Actor to insert minimal highlight tags around pivotal spans while preserving the original context. A frozen Solver then reasons over the emphasized input. The Actor is trained only from the Solver's downstream task reward, requiring no evidence labels, Solver gradients, logits, or internal activations. This yields a solver-compatible alternative to retrieval-style hard selection, context compression, and instance-level prompt rewriting. Across sequential recommendation and QA, HILIGHT consistently improves over manual prompting and strong automated prompt-optimization baselines, with gains up to +27.5% over manual instruction (MI) and +10.8% over the strongest baseline on Amazon-Beauty. The learned emphasis policy transfers zero-shot to smaller and larger unseen Solvers across model families, including API-based Solvers, and its highlights align with human supporting evidence up to 0.78 F1. These results suggest that evidence selection can be learned as a reusable input-side control mechanism for frozen LLMs. Code is available at `https://github.com/yh-sh/HiLight`.

1 Introduction

Large Language Models (LLMs) are increasingly deployed in settings that require reasoning over long and imperfect contexts, such as multi-document question answering [1], agentic workflows with accumulated traces [52, 36], and personalization over noisy user histories [19, 17]. Despite progress in scaling model capacity and context length, effective context utilization remains a bottleneck: LLMs can overlook decisive evidence when it is diluted among irrelevant, redundant, or outdated information [49]. Empirical studies have documented this failure mode, including the "Lost in the Middle" effect, where models attend less reliably to information that is neither near the beginning nor the end of a long input [26]. As a result, simply providing more context can hurt as much as it helps, especially when the context contains many plausible distractors. For example, in sequential recommendation a single preference cue may appear once in a long history, and in multi-hop QA a small bridging fact can be easy to miss even when both supporting passages are present.

A useful perspective is that long-context failures often arise from coupling two subproblems into one forward pass: *evidence selection*, which identifies task-relevant spans amid distractors, and *reasoning*, which performs inference over that evidence. Modern LLMs can be strong reasoners given clean, focused inputs, yet may still fail to isolate sparse evidence in low signal-to-noise regimes. This motivates an explicit decoupling: rather than forcing a single Solver to both sift through noise and

[*]Joint first author. [†]Joint correspondence author.

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

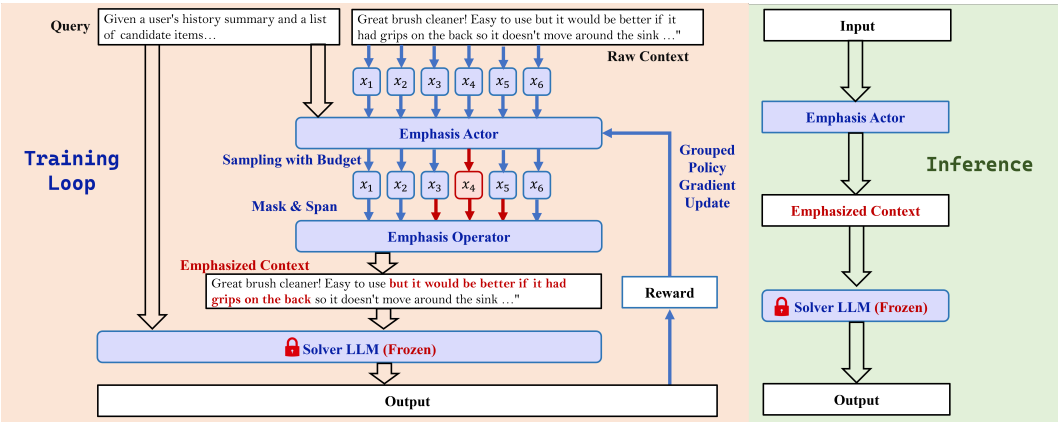


Figure 1: **Overview of HILIGHT.** HILIGHT decouples *evidence selection* from *reasoning* in long, noisy contexts. **Inference:** Given query Q and context X, a lightweight *Emphasis Actor* selects pivotal spans under budget γ and inserts minimal tags to form $\hat{X}$. A frozen *Solver LLM* then produces the output. **Training:** With no evidence annotations, we optimize the Actor using weakly supervised RL from the Solver reward $R(y, y^*)$, without Solver gradients or activations.

reason, we introduce a lightweight module whose sole role is to *surface evidence*, allowing a frozen Solver to focus on reasoning. If this decoupling is successful, the resulting evidence cues should not only help a given Solver, but may also remain useful across *different Solver families and scales*, since they reflect salient structure in the input rather than the idiosyncrasies of a single backbone.

To this end, we propose HILIGHT, an *Evidence Emphasis* framework that learns *instance-wise input markup* for long, noisy contexts. Inspired by how humans mark pivotal fragments when reading dense documents, HILIGHT introduces a lightweight *Emphasis Actor* that inserts minimal structural markers around selected spans in the *original* context. The emphasized context is then consumed by a frozen *Solver LLM* to produce the final answer or prediction (Figure 1). Crucially, HILIGHT does not compress, summarize, or rewrite the input: it preserves the raw text and intervenes only through lightweight, human-readable highlighting.

Thus, HILIGHT studies a distinct design point: learning how to present the same fixed context to a frozen Solver, rather than changing which documents are retrieved, rewriting the context, or modifying the Solver. This matters because many context-management strategies trade fidelity for focus: retrieval and re-ranking select only a subset of passages [53, 61, 38], pruning discards non-selected text [14], compression and summarization may omit subtle evidence [15, 16, 46, 10, 32, 7, 43], and rewriting can alter the original input [42]. HILIGHT instead treats evidence selection as learned, non-destructive input-side control: the Solver still receives the full context, while the Actor learns where to emphasize using only task reward. The central question is therefore not whether less context is better, but whether learned emphasis can improve how a frozen Solver uses the same context.

This design yields three practical benefits. First, it is *solver-compatible*: HILIGHT treats the Solver as frozen and requires only Solver outputs to compute reward, without gradients or activations, making it compatible with open-weight, API-based, or proprietary LLMs. Second, it is *auditable*: the Actor produces explicit highlighted spans that expose what evidence was surfaced. Third, it is *fidelity-preserving*: the original context remains intact, so surrounding information can still support disambiguation and multi-hop reasoning. As we later show, this input-side formulation also enables the learned emphasis policy to *transfer zero-shot across Solver families and scales.*

While the high-level intuition is simple, it raises two tightly coupled questions:

> *How can we learn budgeted, instance-wise evidence highlighting for a frozen LLM using only downstream task feedback? When does non-destructive emphasis help more than hard selection or ordinary prompting?*

We answer the first question with a weakly supervised learning formulation. The key operational challenge is that "important evidence" is typically latent and task dependent: most benchmarks provide only an input and an outcome label, not ground truth token level evidence. We therefore

optimize the Emphasis Actor via reinforcement learning (RL) using only the Solver's downstream task reward. This enables the Actor to learn an evidence selection policy aligned with the task and Solver while requiring neither evidence annotations nor access to Solver internals.

We answer the second question empirically by studying HILIGHT across tasks with different evidence sparsity and distractor structure. Table 1 shows that HILIGHT improves consistently over manual prompting and strong automated prompt-optimization baselines across sequential recommendation, multi-hop QA, reading comprehension, and biomedical QA classification. The gains are largest in high-distractor, sparse-signal settings: on Amazon-Beauty, HILIGHT improves over manual instruction (MI) by +27.5% HR@10 and +27.3% NDCG@10, and over the strongest baseline by +10.8% NDCG@10. Gains are smaller but consistent when evidence is more concentrated and the remaining bottleneck is complex inference. Consistent with this regime view, pruning to Actor-selected spans helps in sparse recommendation but hurts HotpotQA, while HILIGHT improves both by preserving connective context. Additional analyses show that HILIGHT is stable across budgets and marker formats, transfers zero-shot to smaller/larger unseen Solvers including API-based Solvers, and learns highlights that substantially overlap with human-annotated evidence despite using no token-level supervision.

Our main contributions are summarized as follows:

• **Non-destructive input-side control.** We introduce HILIGHT, an Evidence Emphasis framework that treats evidence selection as learned input-side control for frozen LLMs: it surfaces pivotal spans with minimal highlight tags while preserving the full original context.

• **Reward-only Actor learning.** We formulate instance-wise highlighting as a solver-compatible, weakly supervised RL problem and train a lightweight Emphasis Actor using only downstream task reward, without evidence annotations or access to Solver internals.

• **Empirical gains and regime characterization.** Across sequential recommendation and long-context QA benchmarks, HILIGHT consistently outperforms manual prompting and strong prompt-optimization baselines, with the largest gains in high-distractor sparse-evidence regimes. Ablations show that non-destructive emphasis can outperform hard pruning when connective context matters.

• **Transferable and meaningful evidence.** A single Actor transfers zero-shot to smaller and larger unseen Solvers across model families, including API-based Solvers, while its highlights align with human-annotated evidence up to 0.78 F1 despite no token-level supervision.

2 Problem Formulation

We formalize *evidence emphasis* for long-context reasoning with a *frozen* Solver LLM.

Task setup. Let $(Q, X, y^*) \sim \mathcal{D}$ denote a data instance, where Q is a query (or instruction), $X = (x_1, x_2, \ldots, x_L)$ is a tokenized context of length L, y^* is the ground-truth output. A frozen Solver LLM $\mathcal{M}$ maps (Q, X) to an output y (deterministically or stochastically). Performance is measured by a task-specific utility $R(y, y^*)$ such as EM, F1, accuracy, HR@10, or NDCG@10.

Latent evidence. As contexts grow longer, task-relevant information is often diluted by irrelevant, redundant, or outdated content. Conceptually, we can view each instance as containing one or more *latent* evidence subsets within the context. Let $[L] = \{1, 2, \ldots, L\}$ and define an (unobserved) *evidence set*

$$E := \{i \in [L] : x_i \text{ is part of task-relevant evidence for } (Q, X)\}.$$

We emphasize that E is *not* observed during training, need not be unique, and may depend on both the task and the Solver; we assume no token-level evidence annotations.

Evidence highlighting policy. We learn an Actor policy $\pi_\theta(\cdot \mid Q, X)$ that selects which parts of the context to emphasize. For simplicity, we begin with token-level selection via a binary mask $M \in \{0, 1\}^L$, where $M_i = 1$ indicates token x_i is emphasized. Given (X, M), we construct an emphasized context $\hat{X} = g(X, M)$, where g is a deterministic, *non-destructive* operator that inserts minimal highlight tags around selected tokens (or coalesced spans) while preserving the original token content and order.

Highlight budget. To prevent degenerate all-highlight solutions and to keep emphasis concise and auditable, we impose a budget:

$$\sum_{i=1}^{L} M_i \leq \gamma L, \tag{1}$$

where $\gamma \in (0, 1]$ controls the maximum fraction of emphasized tokens. In practice, we enforce this constraint via a projection Proj_k with $k = \lfloor \gamma L \rfloor$.

Objective. For each instance, the Actor first samples a preliminary mask $\tilde{M} \sim \pi_\theta(\cdot \mid Q, X)$, projects it to a feasible mask $M = \mathrm{Proj}_k(\tilde{M})$, and feeds the emphasized context $g(X, M)$ to the frozen Solver. The learning objective is

$$\mathcal{J}(\theta) = \mathbb{E}\left[R\left(y, y^*\right)\right], \quad y \sim \mathcal{M}(\cdot \mid Q, g(X, M)), \tag{2}$$

where the expectation is over $(Q, X, y^*) \sim \mathcal{D}$, $\tilde{M} \sim \pi_\theta(\cdot \mid Q, X)$, the projection $M = \mathrm{Proj}_k(\tilde{M})$, and any stochasticity of $\mathcal{M}$. If the Solver is deterministic, the final source of randomness is absent. Since evidence masks are latent and $\mathcal{M}$ is treated as a black box, $\mathcal{J}(\theta)$ cannot be optimized by backpropagating through the Solver. We therefore optimize the Actor with policy-gradient methods using only the downstream reward $R(y, y^*)$.

3 HILIGHT Method

This section instantiates the formulation in Section 2. Given a frozen Solver LLM $\mathcal{M}$ and a non-destructive emphasis operator $g(X, M)$, we describe (i) how HILIGHT parameterizes the Actor policy $\pi_\theta(M \mid Q, X)$, (ii) how we enforce the highlight budget and construct emphasized inputs via span coalescence and tag injection, and (iii) how we optimize the Actor using task-level rewards.

3.1 Emphasis Actor

Backbone and inputs. The Actor π_θ is a pretrained LM backbone that encodes the concatenated input $[Q; X]$. Let $\mathbf{H} = (\mathbf{h}_1, \ldots, \mathbf{h}_L)$ denote the hidden states for the L context tokens conditioned on Q[1].

Token importance scores. We map each context-token representation to an importance probability:

$$p_i = \sigma\left(\frac{\mathbf{w}_p^\top \mathrm{LayerNorm}(\mathbf{h}_i)}{\tau}\right), \tag{3}$$

where $\mathbf{w}_p \in \mathbb{R}^d$ is learnable, τ is a temperature, and $\sigma(\cdot)$ is the logistic function. This yields a probability map $P = \{p_1, \ldots, p_L\}$ that defines a Bernoulli factorized policy over token selections.

3.2 Budgeted Selection and Emphasis Construction

Training-time sampling with budget projection. The selected mask must satisfy the budget in Eq. (1). During training, we sample a preliminary mask $\tilde{M} \in \{0, 1\}^L$ with $\tilde{m}_i \sim \mathrm{Bernoulli}(p_i)$, then project it to a feasible mask

$$M = \mathrm{Proj}_k(\tilde{M}), \qquad k = \lfloor \gamma L \rfloor, \tag{4}$$

where Proj_k retains at most k selected tokens, prioritizing larger p_i values. This preserves exploration while guaranteeing budget feasibility; the projection is treated as part of the environment transition and is not differentiated through.

Inference-time deterministic top-k. At inference, we use a deterministic top-k policy for stability, setting $M_i = 1$ for the k largest probabilities and 0 otherwise.

Span coalescence and Solver prompting. Given M, the emphasis operator g merges adjacent selected tokens into spans, optionally bridges short gaps of at most δ tokens, and inserts boundary tags

[1]We implement the Actor with full attention over (Q, X) to support evidence selection in long contexts, while the Solver remains frozen and can be open-weight or API-based.

such as <start_important> and <end_important> at span endpoints. The resulting emphasized context $\hat{X} = g(X, M)$ preserves the original text content and order. The frozen Solver receives $(Q, \hat{X})$ under a fixed prompt template that tells it to treat tagged spans as emphasized evidence while still allowing it to use the surrounding context. Since HILIGHT only changes the input string and never accesses Solver gradients, logits, or intermediate activations, it is compatible with black-box and API-based Solvers. Full prompt templates are provided in Appendix B.

3.3 Actor Optimization with Task-level Reward

We freeze $\mathcal{M}$ and optimize only π_θ. Because mask decisions are discrete and task metrics are generally non-differentiable, we train the Actor with a grouped policy-gradient objective using the downstream task metric as reward.

Grouped policy gradient. For each training instance, we sample G preliminary masks $\{\tilde{M}_j\}_{j=1}^{G}$, project each to $M_j = \mathrm{Proj}_k(\tilde{M}_j)$, construct $\hat{X}_j = g(X, M_j)$, query the Solver, and obtain rewards

$$r_j = R(y_j, y^*), \qquad y_j \sim \mathcal{M}(\cdot \mid Q, \hat{X}_j). \tag{5}$$

We normalize rewards within the group to form advantages

$$\hat{A}_j = \frac{r_j - \mu_r}{\sigma_r + \epsilon}, \tag{6}$$

where μ_r and σ_r are the mean and standard deviation of $\{r_j\}_{j=1}^{G}$. The policy-gradient loss is

$$\mathcal{L}_{\mathrm{PG}} = -\frac{1}{G} \sum_{j=1}^{G} \hat{A}_j \log \pi_\theta(\tilde{M}_j \mid Q, X), \tag{7}$$

where the log-probability is computed under the pre-projection Bernoulli policy; the projected masks are used only to construct emphasized inputs and evaluate rewards.

Regularization. To stabilize training, we add two regularizers. First, an entropy bonus encourages exploration and avoids premature collapse:

$$\mathcal{L}_{\mathrm{ENT}} = -\mathcal{H}(\pi_\theta(\cdot \mid Q, X)), \tag{8}$$

where $\mathcal{H}$ is the token-wise Bernoulli entropy averaged over policy-controlled tokens. Second, a soft length regularizer encourages the expected highlight fraction to match the budget:

$$\mathcal{L}_{\mathrm{LEN}} = \left(\frac{1}{|\Omega|} \sum_{i \in \Omega} p_i - \gamma \right)^2, \tag{9}$$

where Ω denotes valid policy-controlled context tokens. The entropy term maintains exploration, while the length term stabilizes the expected selection density before projection.

Final loss. The final objective minimized by the Actor is

$$\mathcal{L}(\theta) = \mathcal{L}_{\mathrm{PG}} + \lambda_{\mathrm{len}} \mathcal{L}_{\mathrm{LEN}} + \beta_{\mathrm{ent}} \mathcal{L}_{\mathrm{ENT}}. \tag{10}$$

We optimize θ with standard first-order optimizers. Appendix D.3 ablates the entropy and length-regularization terms, showing that removing either term degrades performance. Algorithmic details and the exact Solver prompt template are provided in Appendix B.

4 Experiments

We evaluate HILIGHT on long-context tasks spanning different evidence sparsity and distractor regimes. Our experiments ask four questions: (i) Does instance-wise evidence emphasis improve frozen-Solver performance over strong prompt-optimization baselines? (ii) Is *non-destructive* emphasis preferable to hard pruning? (iii) What is the per-query inference and training cost? and (iv) Does the learned emphasis policy transfer across Solver families and produce meaningful evidence highlights?

Baseline	Amazon-Beauty		HotpotQA		SQuAD 2.0		PubMedQA	
	HR@10	NDCG@10	EM	F1	EM	F1	Acc.	M-F1
MI	0.02256	0.01247	0.556	0.706	0.586	0.647	0.921	0.776
PRL	0.02692	0.01432	0.571	0.719	0.644	0.673	0.924	0.735
BFRS	0.02363	0.01248	0.593	0.700	0.573	0.694	0.922	0.761
OPRO	0.02332	0.01346	0.591	0.723	0.620	0.672	0.922	0.768
DSPy (MIPROv2)	0.02730	0.01410	0.565	0.721	0.590	0.685	0.930	0.767
APE	0.02390	0.01311	0.581	0.722	0.592	0.669	0.926	0.727
HiLight	**0.02877**	**0.01587**	**0.606**	**0.741**	**0.661**	**0.721**	**0.940**	**0.821**
$\Delta_{\%}$(vs. Best Base.)	+5.38%	+10.82%	+2.19%	+2.49%	+2.64%	+3.89%	+1.08%	+5.80%
$\Delta_{\%}$(vs. MI)	+27.53%	+27.27%	+8.99%	+4.96%	+12.80%	+11.44%	+2.06%	+5.80%

Table 1: Overall performance comparison. Best results are highlighted in **purple** and second-best results are marked in blue. $\Delta_{\%}$ is the relative improvement $100\,(m_{\text{HiLight}} - m_b)/m_b$ over baseline b. HiLight consistently outperforms all baselines across sequential recommendation, multi-hop QA, reading comprehension, and biomedical QA classification tasks.

4.1 Experimental Setup

Tasks and datasets. We report results on multiple tasks spanning different sparsity regimes. Our evaluation includes Amazon-Beauty [13], a widely used sequential recommendation dataset containing user review histories. For Amazon-Beauty, each example consists of a user history, one held-out ground-truth next item, and a fixed candidate set constructed following the standard evaluation protocol of [13]; HR@10 and NDCG@10 are computed over this candidate set. Details on candidate construction, negative sampling, and prompt formatting are provided in Appendix C. We also evaluate on HotpotQA [50], a dataset designed for multi-hop question answering; SQuAD 2.0 [34], a reading comprehension benchmark that includes unanswerable questions; and the artificial subset of PubMedQA [18], a biomedical dataset where the task is to answer research questions (Yes/No) based on abstracts.

Solvers, Actors, and rewards. Unless otherwise specified, we use Qwen3-14B [47] as the frozen Solver. The Actor is initialized from Qwen3 models: 0.6B for SQuAD 2.0/PubMedQA, 1.7B for Amazon-Beauty, and 4B for HotpotQA. Appendix D.5 provides a controlled Actor-size study, showing that larger Actors help most on structurally complex evidence-selection tasks such as HotpotQA, with diminishing returns beyond mid-sized Actors. The highlight budget is $\gamma = 0.25$ for Amazon-Beauty and $\gamma = 0.15$ for the QA tasks. We optimize with Adam using learning rate 10^{-4}, weight decay 10^{-2}, group size $G = 4$, entropy coefficient $\beta_{\text{ent}} = 1.0$, and length-regularization weight $\lambda_{\text{len}} = 0.01$. Training rewards are task-specific: a weighted HR@10/NDCG@10 reward for Amazon-Beauty, F1/EM for HotpotQA and SQuAD 2.0, and accuracy for PubMedQA; full details are in Appendix D. Each Actor is trained in one pass over 3,000 (Amazon-Beauty) or 10,000 (QA) training examples, with the checkpoint selected on a disjoint 500-example training subset (Appendix F.5).

Baselines. We compare against MI and strong black-box prompt-optimization baselines: PRL [3], BFRS [37], OPRO [48], DSPy/MIPROv2 [33], and APE [63]. All methods use the same frozen Solver and the same raw context. Full prompt templates and configurations are in Appendix F.

4.2 Main Results

Table 1 shows that HiLight achieves the best reported performance across all four tasks and eight metrics, outperforming both manual instruction (MI) and strong prompt-optimization baselines. This supports our central hypothesis: *instance-wise emphasis on the data context* is a complementary and often stronger control surface than optimizing a single system prompt shared across inputs.

Where the gains are largest: sparse evidence under heavy distractors. The largest gains appear on Amazon-Beauty, where user histories are long and noisy and the decisive signal is sparse. HiLight improves over the strongest baseline by +5.38% HR@10 and +10.82% NDCG@10, and over MI by +27.53% HR@10 and +27.27% NDCG@10. This is precisely the regime where evidence selection is the dominant bottleneck, consistent with HiLight's goal of surfacing pivotal spans without rewriting the context. On HotpotQA, SQuAD 2.0, and PubMedQA, gains are smaller but consistent (1.08%–5.80% over the best baseline), suggesting that explicit emphasis can still improve evidence utilization when reasoning remains a substantial part of the task.

Quality of evidence utilization. Across tasks, HILIGHT often yields relatively larger improvements on ranking- and overlap-sensitive metrics (NDCG@10, F1, M-F1) than on coarse binary metrics (HR@10, EM, Acc.). This is consistent with emphasis improving *evidence utilization quality*: in recommendation, surfaced signals can move the ground-truth item upward within the top-K list (reflected by NDCG@10) even when it was already a "hit"; in QA, emphasizing supporting spans can produce more complete answer spans and higher token overlap (reflected by F1/M-F1) even when exact match is unchanged. Thus, HILIGHT *tends to improve how the Solver uses relevant evidence, not only whether a subset of outputs flip from incorrect to correct.*

4.3 Why Non-Destructive Emphasis?

To isolate the role of non-destructive emphasis, Table 2 compares HILIGHT with four controls: MI (no highlighting), Random highlighting under the same budget, Few-shot highlighting demonstrations, and Pruned, which feeds only Actor-selected spans to the Solver while discarding the rest.

Sparse-evidence recommendation. On Amazon-Beauty, Random highlighting slightly improves over MI, and few-shot exemplars improve further, consistent with the idea that *long noisy histories benefit from explicit focus cues that help the Solver focus on a subset of relevant signals.* The Pruned variant is strong (0.02740 HR@10, 0.01448 NDCG@10), showing that restricting the Solver to the Actor-selected spans can already recover much of the predictive signal in this sparse-evidence regime. However, HILIGHT performs best, indicating that *non-destructive emphasis* is advantageous: *highlighting steers the Solver toward key spans while preserving surrounding context that can help with disambiguation and preference consistency.*

Context preservation for reasoning. A contrasting pattern appears on HotpotQA: Random highlighting, Few-shot highlighting, and especially Pruned context do not improve over MI, and pruning substantially hurts EM. This suggests that multi-hop QA is sensitive to preserving *connective* material beyond the final supporting facts, such as bridging entities, discourse cues, and local context that help the Solver link evidence across passages. HILIGHT is the only variant that improves EM and yields the strongest overall gains, supporting our design choice to *emphasize without deleting*: focus cues help, but removing surrounding context can undermine multi-step reasoning even when the selected spans are informative.

Variant	Amazon-Beauty		HotpotQA	
	HR@10	NDCG@10	EM	F1
MI	0.02256	0.01247	0.556	0.706
Random	0.02380	0.01270	0.521	0.635
Few-shot	0.02403	0.01315	0.556	0.695
Pruned	0.02740	0.01448	0.525	0.658
HILIGHT	**0.02877**	**0.01587**	**0.606**	**0.741**

Table 2: Ablation study on Amazon-Beauty and HotpotQA tasks. Best results are highlighted in **purple** and second-best results are marked in blue.

Relation to within-context retrieval. This ablation also clarifies the relation to retrieval-style baselines in a fixed-context setting. Once the full context is already provided to the Solver, chunk retrieval or reranking within that context reduces to *hard selection*: scoring chunks or spans, retaining the top-k, and discarding the rest. The Pruned ablation directly probes this family. Results show that hard selection can help when the signal is sparse and distractor-heavy, but can hurt reasoning-heavy QA when connective context is removed. This motivates HILIGHT's non-destructive design. Hard selection remains below HILIGHT on HotpotQA even when the selector is not the HILIGHT Actor: an off-the-shelf reranker that keeps the top-k sentences reaches 0.568/0.710 EM/F1, and a pruning selector trained with the same Actor architecture, RL procedure, reward, and Solver-query budget reaches 0.563/0.707, compared with 0.606/0.741 for HILIGHT (Appendix D.9).

4.4 Long-Context Stress Test

To test whether HILIGHT remains useful as distractor load increases, we extend HotpotQA inputs by injecting distractor documents to create contexts of approximately 8K, 16K, and 32K tokens. Table 3 reports relative gains over MI at each length; full absolute EM/F1 values are provided in Appendix D.5. HILIGHT maintains positive gains throughout, supporting the view that the method targets evidence sparsity under distractors rather than only a particular benchmark length.

Context length	∼2K	∼8K	∼16K	∼32K
Δ EM	+9.0%	+5.0%	+4.6%	+3.9%
Δ F1	+5.0%	+4.0%	+2.7%	+4.0%

Table 3: HotpotQA stress test with injected distractors. Values are relative gains over MI; full results are in Appendix D.5.

Method	MI	PRL	BFRS	DSPy (MIPROv2)	APE	OPRO	Few-shot with highlights	HILIGHT
Input tokens	1.0×	1.0×	3.0×	1.0×	1.0×	1.0×	3.0×	$< 1.01\times$
Training Solver queries	0	120K	60K	24K	60K	30K	0	12K

Table 4: Efficiency and Solver cost on Amazon-Beauty. Input tokens are normalized to MI; training cost counts Solver queries.

4.5 Efficiency: Tokens, Solver Queries, and Latency

Table 4 reports efficiency on Amazon-Beauty. We measure *Solver-side cost* in two complementary ways: (i) input-token overhead at inference, and (ii) the number of *Solver queries* used during optimization, which is the dominant shared expense across black-box prompt-optimization methods.

Minimal Solver-token overhead at inference. At inference, HILIGHT uses one lightweight Actor pass followed by a single Solver call per instance. Because HILIGHT only inserts short boundary markers around selected spans, the Solver input length increases by less than $1.01\times$ while preserving the full raw context. In contrast, in-context baselines that rely on demonstrations (e.g., BFRS and Few-shot) expand the Solver input to $3.0\times$, directly reducing the usable context window for long histories. Thus, HILIGHT improves context utilization without paying the large input-length cost of demonstration-based prompting.

Favorable accuracy–Solver-query trade-off. Search-based prompt optimizers spend many Solver calls evaluating candidate instructions, demonstrations, or prompt programs. HILIGHT instead uses Solver calls to train a token-level evidence selector and reaches strong performance with 12K Solver queries, compared with 24K for DSPy, 30K for OPRO, 60K for APE/BFRS, and 120K for PRL. On Amazon-Beauty, this corresponds to a ~ 5–$10\times$ lower Solver-query budget than APE/PRL while also achieving higher accuracy, placing HILIGHT on a favorable accuracy–Solver-cost frontier.

Wall-clock latency is dominated by the Solver. We additionally measure end-to-end wall-clock latency. Across Actor sizes, p95 Actor latency ranges from 0.057 s (0.6B) to 0.635 s (8B), while the 14B Solver takes 9.1–19.6 s depending on task and context length. The 4B Actor used in our most demanding setting adds under 0.34 s at p95, corresponding to roughly 1.3%–2.9% of total pipeline latency. Full latency breakdowns are provided in Appendix D.4.

Scope of the efficiency claim. Our token and latency measurements are for *per-query* fixed-context inference, where each query induces one Actor pass and one Solver call. Conversational multi-turn serving with cache-aware reuse across changing query-conditioned highlights would require an additional systems layer and is left to future work.

4.6 Transferability across Solver Families and Scales

A practical and conceptually important question is whether the evidence-selection policy learned by the Actor is tied to the particular Solver used during training, or whether it remains useful when deployed with substantially different Solvers. This question directly tests the premise behind HILIGHT: if evidence selection can be partly decoupled from reasoning, then the learned emphasis policy should capture reusable structure in the input rather than only the idiosyncrasies of one Solver.

Zero-shot transfer setup. We train the Emphasis Actor once with a *source* Solver, Qwen3-14B, and apply it *zero-shot* to five unseen *target* Solvers spanning different families and scales: Qwen3-4B, Qwen3-8B, Qwen3-32B, Gemma-3-27B, and Llama-3-70B. Neither the Actor nor the target Solver is updated. For each target Solver, we compare against that Solver's own MI baseline, so the reported gains reflect transfer relative to the target model's native performance.

Metric	Method	Smaller than Source		Larger than Source		
		Qwen3-4B	Qwen3-8B	Gemma-3-27B	Qwen3-32B	Llama-3-70B
HR@10	Self-Mark	+3.12%	+3.81%	+7.86%	+5.13%	+6.50%
	HiLight	+13.06%	+13.67%	+15.77%	+8.78%	+19.17%
NDCG@10	Self-Mark	+5.23%	+0.45%	+7.48%	+6.97%	+9.12%
	HiLight	+8.48%	+12.43%	+31.56%	+21.24%	+27.37%

Table 5: Cross-Solver transfer on Amazon-Beauty ($\Delta_{\%}$ over each target Solver's own MI baseline). Actor trained with Qwen3-14B and applied zero-shot.

Transfer across both smaller and larger Solvers. Table 5 shows consistent gains across all five target Solvers, including both smaller and larger models than the training Solver. HiLight improves HR@10 by +8.78% to +19.17% and NDCG@10 by +8.48% to +31.56% relative to each target Solver's own MI baseline. The gains on Llama-3-70B and Gemma-3-27B are especially notable: even substantially larger Solvers, which should already be stronger at filtering distractors, still benefit from explicit evidence emphasis. This suggests that the bottleneck is not only raw reasoning capacity, but also reliable localization and use of sparse evidence in noisy context.

Comparison with self-generated highlighting. We further compare against Self-Mark, where the *target* Solver is prompted to first highlight evidence in the context and then answer using its own self-generated markup. Although Self-Mark gives the target Solver an additional self-highlighting step, HiLight outperforms it across all target Solvers and both metrics. This indicates that a specialized evidence selector trained from task-level feedback can produce higher-quality and more consistent emphasis cues than zero-shot self-highlighting, even when the target Solver is much larger than the Actor-training Solver. In particular, Self-Mark yields weaker and less stable NDCG@10 gains (e.g., only +0.45% on Qwen3-8B), whereas HiLight consistently improves ranking quality. We attribute this gap to the fact that self-highlighting relies on the target Solver's generic instruction-following, which may not align with downstream task reward, while HiLight is trained to select spans that are actually useful for that reward. The larger and more stable NDCG@10 gains further echo Table 1: learned emphasis improves ranking quality, not merely binary hit-or-miss retrieval.

Tokenizer-independent transfer via text spans. Cross-Solver transfer does not require vocabulary or token alignment between the Actor and the target Solver. The Actor's token-level scores are projected back to character-level spans using tokenizer offset mappings. Boundary markers are then inserted into the raw text string, and the target Solver tokenizes this modified text using its own tokenizer. Thus, the transferred object is the emphasized *text span*, not the Actor's token IDs. This also explains why HiLight is compatible with Solvers from different model families and why marker-format robustness in Appendix D.5 supports the claim that gains are not tied to a brittle delimiter tokenization.

Metric	GPT-5 mini	GPT-5 (low)	GPT-5 (high)	Claude Haiku 4.5	Claude Sonnet 5	Claude Opus 4.8
EM	+1.70%	+0.59%	+0.29%	+17.78%	+2.17%	+1.47%
F1	+1.11%	+1.00%	+0.98%	+19.39%	+1.76%	+1.57%

Table 6: Zero-shot transfer to API-based Solvers on HotpotQA ($\Delta_{\%}$ over each Solver's own MI baseline).

Transfer to API-based Solvers and across tasks. Without adaptation, the same Actor improves EM and F1 for all six API-based Solvers in Table 6, with smaller gains for stronger Solvers within each family (Appendix D.6). The HotpotQA Actor applied unchanged to SQuAD 2.0 also improves EM by +3.58% and F1 by +4.64% over MI, compared with +12.80% and +11.44% for the task-specific Actor (Appendix D.10).

Takeaway. Overall, these transfer results support a key premise of HiLight: a substantial portion of evidence-selection knowledge reflects task and input structure rather than only the idiosyncrasies of a single Solver. The learned emphasis policy transfers to smaller Solvers, larger Solvers, different model families, API-based Solvers, and a related task, and Appendix D.5 provides robustness checks

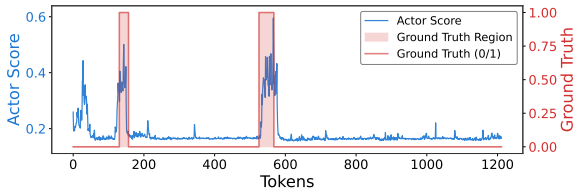


Figure 2: Token-level Actor importance scores on a HotpotQA example. Higher scores concentrate on a small set of regions, with substantial overlap with annotated evidence spans.

Actor Size	Precision	Recall	F1
0.6B	0.7275	0.6433	0.6828
1.7B	0.7964	0.7016	0.7460
4B	0.8367	0.7215	0.7748
8B	0.8403	0.7307	0.7817

Table 7: Token-level overlap between Actor highlights and ground-truth supporting facts on HotpotQA. The Actor is trained without evidence annotations.

across highlight budgets, delimiter formats, span-bridging thresholds, loss coefficients, and Solver prompt templates. Together, these results empirically support the view that evidence selection and reasoning are at least partially separable.

4.7 Evidence Quality and Interpretability

A natural question is whether the learned highlights correspond to semantically meaningful evidence or merely exploit Solver-specific formatting biases. HotpotQA provides human-annotated supporting facts, allowing both qualitative and quantitative evaluation of evidence quality.

Actor scores concentrate on supporting evidence. Figure 2 visualizes token-level Actor scores on a representative HotpotQA instance. The scores form a small number of pronounced peaks, with high-scoring regions substantially overlapping the annotated supporting evidence. Between these peaks, scores remain low, indicating that the Actor uses its budget to focus on a limited set of candidate evidence spans rather than broadly reweighting the entire input.

Quantitative evidence alignment. We threshold Actor scores at 0.5 to obtain predicted evidence tokens and compute precision, recall, and F1 against HotpotQA supporting-fact annotations. Table 7 shows that the Actor achieves up to 0.78 F1 overlap with human evidence, despite never seeing evidence labels during training. The monotonic improvement from 0.68 F1 with a 0.6B Actor to 0.78 F1 with an 8B Actor mirrors the downstream scaling trend, suggesting that additional Actor capacity improves evidence identification rather than merely amplifying formatting effects.

Case studies and failure analysis are provided in Appendix D.8.

5 Conclusion

We introduced HILIGHT, a learned evidence-emphasis framework that treats evidence selection as non-destructive input-side control for frozen LLMs. A lightweight Actor learns from task-level reward alone to insert minimal highlight markers into the original context, allowing a frozen Solver to reason over emphasized but unaltered input. Experiments across recommendation and QA show consistent gains over manual prompting and strong prompt-optimization baselines, especially in high-distractor regimes where evidence is sparse. Transfer and evidence-alignment analyses further suggest that the Actor learns reusable, human-meaningful evidence structure rather than merely exploiting a single Solver's formatting preferences. Our current study focuses on per-query fixed-context inference; cache-aware multi-turn serving, instance-adaptive highlight budgets, and broader rewriting-based comparisons remain important future directions. From a broader-impact perspective, HILIGHT may improve the auditability and reliability of LLM-based decision pipelines by exposing which evidence was emphasized, but incorrect or biased emphasis could also mislead users in sensitive applications.

Limitations. HILIGHT assumes that the query and the context arrive as separate fields; the Actor conditions on both but highlights only the context, and in multi-turn use the highlights must be recomputed whenever either changes. Each task uses its own Actor trained from task reward, a per-task cost that cross-Solver and cross-task transfer reduce but do not remove, and all Actors in our study come from the Qwen3 family. Highlights can also fall on non-decisive spans, and correctly highlighted evidence does not guarantee a correct answer (Appendix D.8). Finally, our bootstrap intervals quantify aggregate sampling uncertainty rather than per-instance calibration, so a highlight is not a confidence estimate for an individual prediction.

Acknowledgements

The work of S. Li, Y. Shi and J. Li was supported in part by the National Science Foundation (NSF) grants 2148309, 2315614 and 2337914, and the National Institutes of Health (NIH) grant R01HL184139. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[3] Paweł Batorski, Adrian Kosmala, and Paul Swoboda. PRL: Prompts from reinforcement learning. *arXiv preprint arXiv:2505.14412*, 2025.

[4] Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Prompting is programming: A query language for large language models. *Proceedings of the ACM on Programming Languages*, 7 (PLDI):1946–1969, 2023.

[5] Jiamin Chen, Yuchen Li, Xinyu Ma, Xinran Chen, Xiaokun Zhang, Shuaiqiang Wang, Chen Ma, and Dawei Yin. Grounding long-context reasoning with contextual normalization for retrieval-augmented generation. *arXiv preprint arXiv:2510.13191*, 2025.

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

[7] Weizhi Fei, Xueyan Niu, Guoqing Xie, Yingqing Liu, Bo Bai, and Wei Han. Efficient prompt compression with evaluator heads for long-context transformer inference. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[8] Chrisantha Fernando, Dylan Sunil Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. Promptbreeder: Self-referential self-improvement via prompt evolution. In *Proceedings of the 41st International Conference on Machine Learning*, pages 13481–13544, 2024.

[9] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *CoRR*, abs/2303.14524, 2023.

[10] Tao Ge, Hu Jing, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024.

[11] 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 *RecSys '22: Sixteenth ACM Conference on Recommender Systems*, pages 299–315, 2022.

[12] Danny Halawi, Jean-Stanislas Denain, and Jacob Steinhardt. Overthinking the truth: Understanding how language models process false demonstrations. In *The Twelfth International Conference on Learning Representations*, 2024.

[13] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*, 2024.

[14] Guang Huang, Yanan Xiao, Lu Jiang, Minghao Yin, and Pengyang Wang. Beyond prompt engineering: a reinforced token-level input refinement for large language models. In *Proceedings of the AAAI conference on artificial intelligence*, pages 24113–24121, 2025.

[15] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376. Association for Computational Linguistics, 2023.

[16] Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LongLLMlingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 1658–1677. Association for Computational Linguistics, 2024.

[17] Yangqin Jiang, Xubin Ren, Lianghao Xia, Da Luo, Kangyi Lin, and Chao Huang. RecGPT: A foundation model for sequential recommendation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 10129–10143, 2025.

[18] Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. 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)*, pages 2567–2577, 2019.

[19] Genki Kusano, Kosuke Akimoto, and Kunihiro Takeoka. Revisiting prompt engineering: A comprehensive evaluation for llm-based personalized recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems, RecSys 2025*, pages 832–841, 2025.

[20] Weixian Waylon Li, Yuchen Niu, Yongxin Yang, Keshuang Li, Tiejun Ma, and Shay B. Cohen. Spectral attention steering for prompt highlighting. In *The Fourteenth International Conference on Learning Representations*, 2026.

[21] Zekun Li, Baolin Peng, Pengcheng He, Michel Galley, Jianfeng Gao, and Xifeng Yan. Guiding large language models via directional stimulus prompting. *Advances in Neural Information Processing Systems*, 36:62630–62656, 2023.

[22] Mingfu Liang, Yufei Li, Jay Xu, Kavosh Asadi, Xi Liu, Shuo Gu, Kaushik Rangadurai, Frank Shyu, Shuaiwen Wang, Song Yang, et al. Generative reasoning re-ranker. *arXiv preprint arXiv:2602.07774*, 2026.

[23] Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, yelong shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Not all tokens are what you need for pretraining. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[24] Aiwei Liu, Haoping Bai, Zhiyun Lu, Yanchao Sun, Xiang Kong, Xiaoming Simon Wang, Jiulong Shan, Albin Madappally Jose, Xiaojiang Liu, Lijie Wen, Philip S. Yu, and Meng Cao. TIS-DPO: Token-level importance sampling for direct preference optimization with estimated weights. In *The Thirteenth International Conference on Learning Representations*, 2025.

[25] Jiahao Liu, Xueshuo Yan, Dongsheng Li, Guangping Zhang, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. Improving llm-powered recommendations with personalized information. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2560–2565, 2025.

[26] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[27] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

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

[29] Zhining Liu, Rana Ali Amjad, Ravinarayana Adkathimar, Tianxin Wei, and Hanghang Tong. SelfElicit: Your language model secretly knows where is the relevant evidence. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9153–9173, 2025.

[30] Youmi Ma and Naoaki Okazaki. From interpretability to performance: Optimizing retrieval heads for long-context language models. *arXiv preprint arXiv:2601.11020*, 2026.

[31] Wenyu Mao, Jiancan Wu, Weijian Chen, Chongming Gao, Xiang Wang, and Xiangnan He. Reinforced prompt personalization for recommendation with large language models. *ACM Transactions on Information Systems*, 43(3):1–27, 2025.

[32] Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[33] Krista Opsahl-Ong, Michael J. Ryan, Josh Purtell, David Broman, Christopher Potts, Matei Zaharia, and Omar Khattab. Optimizing instructions and demonstrations for multi-stage language model programs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pages 9340–9366, 2024.

[34] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, pages 784–789, 2018.

[35] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36: 68539–68551, 2023.

[36] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 8634–8652, 2023.

[37] Dilara Soylu, Christopher Potts, and Omar Khattab. Fine-tuning and prompt optimization: Two great steps that work better together. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10696–10710, 2024.

[38] Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. DynamicRAG: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[39] Yuan Tian and Tianyi Zhang. Selective prompt anchoring for code generation. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008, 2017.

[41] Jianyu Wang, Zhiqiang Hu, and Lidong Bing. Evolving prompts in-context: An open-ended, self-replicating perspective. In *Forty-second International Conference on Machine Learning*, 2025.

[42] Jason Weston and Sainbayar Sukhbaatar. System 2 attention (is something you might need too). *arXiv preprint arXiv:2311.11829*, 2023.

[43] David Wingate, Mohammad Shoeybi, and Taylor Sorensen. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5621–5634, 2022.

[44] Yongliang Wu, Yizhou Zhou, Zhou Ziheng, Yingzhe Peng, Xinyu Ye, Xinting Hu, Wenbo Zhu, Lu Qi, Ming-Hsuan Yang, and Xu Yang. On the generalization of SFT: A reinforcement learning perspective with reward rectification. *arXiv preprint arXiv:2508.05629*, 2025.

[45] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

[46] Fangyuan Xu, Weijia Shi, and Eunsol Choi. RECOMP: Improving retrieval-augmented LMs with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*, 2024.

[47] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[48] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

[49] Minglai Yang, Ethan Huang, Liang Zhang, Mihai Surdeanu, William Yang Wang, and Liangming Pan. How is LLM reasoning distracted by irrelevant context? an analysis using a controlled benchmark. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 13329–13347, 2025.

[50] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

[51] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J. Smola, and Eduard H. Hovy. Hierarchical attention networks for document classification. In *NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

[52] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[53] Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. RankRAG: Unifying context ranking with retrieval-augmented generation in LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[54] Mert Yuksekgonul, Varun Chandrasekaran, Erik Jones, Suriya Gunasekar, Ranjita Naik, Hamid Palangi, Ece Kamar, and Besmira Nushi. Attention satisfies: A constraint-satisfaction lens on factual errors of language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[55] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative ai by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.

[56] Qingru Zhang, Chandan Singh, Liyuan Liu, Xiaodong Liu, Bin Yu, Jianfeng Gao, and Tuo Zhao. Tell your model where to attend: Post-hoc attention steering for LLMs. In *The Twelfth International Conference on Learning Representations*, 2024.

[57] Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E. Gonzalez. TEMPERA: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[58] Yihua Zhang, Xi Liu, Xihuan Zeng, Mingfu Liang, Jiyan Yang, Rong Jin, Wen-Yen Chen, Yiping Han, Hao Ma, Bo Long, Huayu Li, Buyun Zhang, Liang Luo, Sijia Liu, and Tianlong Chen. Reasonrec: A reasoning-augmented multimodal agent for unified recommendation. In *ICML 2025 Workshop on Programmatic Representations for Agent Learning*, 2025.

[59] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36*, pages 34661–34710, 2023.

[60] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. In *The Eleventh International Conference on Learning Representations*, 2023.

[61] Qingfei Zhao, Ruobing Wang, Yukuo Cen, Daren Zha, Shicheng Tan, Yuxiao Dong, and Jie Tang. LongRAG: A dual-perspective retrieval-augmented generation paradigm for long-context question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22600–22632, 2024.

[62] Yuyue Zhao, Jiancan Wu, Xiang Wang, Wei Tang, Dingxian Wang, and Maarten De Rijke. Let me do it for you: Towards llm empowered recommendation via tool learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1796–1806, 2024.

[63] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023.

A Related Work

Long-context utilization and evidence selection. A central challenge in long-context LLM reasoning is to ensure that task-relevant evidence is not diluted by distractors. Most prior approaches manage long inputs by either reducing input volume or explicitly selecting salient regions. A dominant line is *hard selection*, such as retrieving top-k passages, pruning history, or re-ranking candidate context segments before inference [53, 61, 38, 14]. While effective, hard selection can discard surrounding text that is useful for disambiguation, negative reasoning, or maintaining discourse structure. In our setting, the full context is already given to the Solver, so a retrieval-style method within a single context reduces to *hard selection*: chunking the context, scoring chunks or spans, and retaining only the top-k subset while discarding the rest. We study this family through our *Pruned* ablation, which shows that deletion-based selection can help in sparse-signal recommendation but can hurt multi-hop QA when connective context is removed.

Another line reduces context length through prompt compression, learned gist tokens, summarization, or context distillation [15, 16, 32, 46, 10, 7, 43]. These methods reduce length and can increase the density of relevant information, but they are inherently lossy and can introduce errors when the compressed representation omits subtle but decisive details. A related strategy prompts the LLM itself to extract or regenerate a distilled context before answering [42], but this adds an extra generation step, still risks discarding bridging information, and couples evidence selection to the Solver's own instruction-following ability. Most relevant to our setting are methods that extract evidence or rationales from LLM internal signals, such as attention or attribution maps, to support interpretability or post-hoc explanation [54, 12, 29]. For example, SelfElicit [29] highlights evidence at inference time using the Solver's own attention-derived signals, tying evidence selection to the target model's internals. Attention-steering methods share the goal of directing a model toward important input content: PASTA [56] reweights attention toward user-specified spans in selected heads, Selective Prompt Anchoring [39] amplifies the influence of selected prompt tokens on the output logits, and SEKA [20] edits key embeddings to steer attention toward highlighted text. These methods operate on attention or logits inside the model and take the emphasized spans as given, whereas HILIGHT learns where to place emphasis from task reward. In contrast to approaches that require internal access or primarily analyze a fixed model after the fact, HILIGHT performs an *input-side intervention* that is compatible with frozen Solvers: it inserts explicit emphasis markers *before* inference to steer the Solver toward pivotal spans while keeping the original context intact.

Context markup and inference-time control. A related and widely used practice for controlling LLM behavior is to introduce structured markup in the input, such as delimiters, section headers, XML wrappers, and other formatting conventions that shape how the model parses and attends to context [52, 4]. Such structured prompting can improve reliability by making evidence boundaries explicit and by separating instructions from data [52, 35]. Recent context-formatting work such as C-NORM further shows that context presentation choices, including delimiter and structural-marker conventions, can substantially affect long-context utilization [5]. However, such methods typically optimize global or model-specific formatting choices, and they do not address the key challenge in long-context settings: deciding *where* the decisive evidence is for each instance [21]. HILIGHT bridges this gap by learning *instance-wise* placement of minimal highlight tags on top of an unaltered context, effectively providing a learned control surface for evidence emphasis. Orthogonally, model-internal approaches such as RetMask improve context utilization by modifying or post-training the Solver [30], whereas HILIGHT changes only the input string and remains compatible with frozen and API-based Solvers. Our robustness experiments further test whether gains persist across different tag formats and highlighting budgets, indicating that the method is not merely exploiting a brittle formatting quirk.

Automatic Prompt Optimization (APO) and test-time scaling. Traditional prompt engineering relies on manual trial-and-error [27], while APO automates this process by treating prompts as learnable objects [60]. Existing methods optimize instructions via gradient-based updates [55], evolutionary search [8, 41], or RL [57, 31]. A fundamental distinction is the optimization scope: APO typically searches for a single, static system prompt optimized at the *task level*. In contrast, HILIGHT performs *instance-level* optimization on the *data context* itself by dynamically deciding where to place saliency markers for each specific input, enabling adaptation to varying noise and evidence distributions across queries while consuming the same raw evidence. Orthogonally, test-time scaling strategies such as best-of-N sampling or self-consistency can improve accuracy but often increase

latency and compute; HILIGHT instead aims to improve evidence utilization via a single emphasized input, without relying on extensive over-sampling.

Token-level optimization. The axiom that tokens possess unequal semantic weight is foundational to modern NLP, underpinning mechanisms from attention [2, 51, 40] to efficient inference [59, 45]. Recent works exploit token non-uniformity to refine training objectives, reweight losses, or optimize token-level inputs [23, 24, 44, 14]. Closest in spirit, reinforced token-level input refinement learns token-level edits or selections through RL, but its goal is input refinement rather than non-destructive evidence emphasis for a frozen Solver; refinement or pruning can change what information is passed forward. HILIGHT instead externalizes token importance as explicit structural markers: it inserts emphasis cues into the raw input, preserves all original tokens, and trains a separate Actor from task-level reward without accessing or modifying the Solver.

LLMs for recommendation and personalization. Recent work has explored using LLMs as recommender components, including zero-shot or few-shot recommendation via natural-language prompts [9, 19, 25, 17], modeling user histories and item identifiers in semantic space [11, 6, 28, 22], and hybrid systems that integrate LLM representations with classic recommenders and external tools [62, 58]. Despite promising results, LLM-based recommendation faces an extreme long-context regime: user histories can be ultra-long, noisy, and temporally non-stationary, so relevant signals are often sparse and easily overshadowed by distractors. This makes recommendation a natural stress test for evidence emphasis, and our results show that HILIGHT provides a lightweight and interpretable mechanism for improving long-context utilization in such noisy personalization settings.

B Training Algorithm and Solver Prompt Template

B.1 Frozen Solver Prompting

[CONTEXT]
`... the patient exhibited <start_important> symptoms of severe fatigue <end_important> ...`
(The `<start_important>...<end_important>` tags indicate emphasized evidence.)

[INSTRUCTION]
`Answer the query using the context above.`

Figure 3: The consistent prompt template used for the frozen Solver. The Emphasis Actor injects blue tags into the raw context.

The Solver LLM $\mathcal{M}$ receives $(Q, \hat{X})$ and generates an output y. We use a consistent prompt template that encourages attention to highlighted spans while still allowing the model to use surrounding context when needed. Because HILIGHT only modifies inputs and never requires gradients or intermediate activations from $\mathcal{M}$, it is compatible with both open-weight and API-based Solvers.

B.2 Actor Optimization with Task-level Reward

Algorithm 1 summarizes the complete training procedure for HILIGHT, including grouped sampling, budget projection, emphasis construction, and the Actor update under Eq. (10).

C Benchmarks

To evaluate the effectiveness of our framework, we compare against a broad suite of baselines spanning manual engineering and state-of-the-art automated optimization methods:

- **MI** (Manual Instruction). A foundational baseline in which the Solver LLM is prompted with human-written instructions based on intuition, without automated optimization or any structural emphasis applied to the context.

Algorithm 1 HILIGHT training with grouped policy gradient (Solver frozen)

Input: Dataset $\mathcal{D}$, Actor π_θ, frozen Solver $\mathcal{M}$, group size G, budget γ.
while not converged **do**
  Sample a batch $B = \{(Q, X, y^*)\} \sim \mathcal{D}$.
  for each (Q, X, y^*) in B **do**
    for $j = 1$ to G **do**
      Compute $P = \{p_i\}$.
      Sample $\tilde{M}_j$ via Bernoulli(p_i) and project to budget $k = \lfloor \gamma L \rfloor$ to obtain M_j.
      Construct $\hat{X}_j = g(X, M_j)$ (span coalescence + tag insertion).
      Query Solver $y_j \leftarrow \mathcal{M}(Q, \hat{X}_j)$ and compute reward $r_j \leftarrow R(y_j, y^*)$.
    end for
    Compute $\{\hat{A}_j\}$ by normalizing $\{r_j\}$ within the group (Eq. (6)).
    Update Actor parameters θ by minimizing Eq. (10).
  end for
end while
Output: Optimized Actor π_θ.

- **PRL** [3]. A reinforcement learning–based framework that trains a prompt generator to iteratively improve instructions by producing an intermediate reasoning trace before emitting the final prompt, and autonomously synthesizes and integrates task-specific few-shot examples to boost the performance of a frozen evaluation model. *LLM-call cost:* with 3,000 training datapoints, PRL typically requires (i) running the evaluation LLM to obtain rewards/scores and (ii) additional generator calls to sample candidate prompts (often with multiple rollouts per update). These repeated score–and–update cycles over the training set (and sometimes across multiple epochs/mini-batches) lead to the largest total number of calls.
- **BFRS** [37]. An optimization strategy for modular language-model pipelines that performs best-first search over prompt candidates and iteratively refines them using feedback from model executions. By bootstrapping training data from the model's own successful executions, BFRS enables self-improvement. *LLM-call cost:* BFRS expands a search tree of candidates; each expansion requires scoring many nearby prompt variants on a subset of the 3,000 examples, and it additionally spends calls on generating new candidates (mutations) from the current best prompts. Although it can prune aggressively, the repeated evaluate–expand loop still results in a multi-$\times$ overhead in calls compared with single-pass induction, consistent with the $\sim 3\times$ cost in the table.
- **OPRO** [48]. A method that uses LLMs as optimizers by specifying the optimization objective in natural language. In an iterative loop, the LLM proposes new candidates from a meta-prompt containing prior candidates and their scores, progressively improving performance over time. *LLM-call cost:* each OPRO iteration uses (i) one (or a few) optimizer-LLM calls to propose a batch of new instructions and (ii) many evaluation-LLM calls to score those candidates on the 3,000 training examples (often via a fixed evaluation subset per iteration). Total calls therefore scale roughly with #iterations$\times$#candidates$\times$#eval-examples, yielding the moderate budget shown in the table.
- **DSPy (MIPROv2)** [33]. An optimizer for multi-stage language-model programs that jointly tunes free-form instructions to maximize task performance without requiring module-level labels. It uses a grounded proposer to generate high-quality candidates and a Bayesian surrogate model to efficiently search the combinatorial space of prompt configurations. *LLM-call cost:* DSPy amortizes calls by (i) proposing relatively few high-quality candidates per round and (ii) using a surrogate model to prioritize which configurations to evaluate, so it does not need to exhaustively score every candidate on all 3,000 datapoints. As a result, the dominant cost is a limited number of evaluation runs (plus occasional proposer calls), leading to the smallest overall call budget in our setting.
- **APE** [63]. A method that frames instruction generation as natural-language program synthesis: given input–output demonstrations, an LLM generates candidate prompts, which are then scored and filtered to select the instruction that best elicits the desired behavior. *LLM-call cost:* APE requires (i) generation calls to synthesize a pool of candidate instructions from demonstrations and (ii) evaluation calls to score each candidate on a subset (or all)

of the 3,000 training datapoints. Since the main loop is "generate a batch, then score and select," the total cost is dominated by candidate scoring and scales with #candidates×#eval-examples.

D Additional Experimental Details and Results

D.1 Training Rewards and Hyperparameters

Table 8 specifies the training reward $R(y, y^*)$ used for each task. We use composite rewards when the primary metric is too sparse for stable policy-gradient optimization.

Task	**Training Reward** $R(y, y^*)$
Amazon-Beauty	$0.7 \times \text{HR@10} + 0.3 \times \text{NDCG@10}$
HotpotQA	$0.5 \times \text{F1} + 0.5 \times \text{EM}$
SQuAD 2.0	$0.5 \times \text{F1} + 0.5 \times \text{EM}$
PubMedQA	Accuracy (Yes/No)

Table 8: Training reward used for each task.

Unless otherwise specified, we use Adam with learning rate 10^{-4}, weight decay 10^{-2}, group size $G = 4$, entropy coefficient $\beta_{\text{ent}} = 1.0$, and length-regularization weight $\lambda_{\text{len}} = 0.01$. The default highlight budget is $\gamma = 0.25$ on Amazon-Beauty and $\gamma = 0.15$ on the QA tasks. All experiments run on a node with four NVIDIA RTX PRO 6000 GPUs.

D.2 Span Coalescence and Sampling Strategy

Table 9 ablates the span-coalescence gap width δ and the test-time sampling method on Amazon-Beauty. Performance is stable across a broad range of widths and sampling strategies.

Width	**Greedy-TopK**		**Softmax**		**Gumbel-TopK**	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
4	0.0271	0.0150	0.0271	0.0147	0.0266	0.0147
6	0.0270	0.0152	0.0273	0.0150	0.0270	0.0152
8	0.0283	0.0157	0.0277	0.0153	0.0273	0.0152
10	0.0288	0.0159	0.0282	0.0155	0.0270	0.0150
12	0.0281	0.0158	0.0260	0.0149	0.0275	0.0153
14	0.0271	0.0154	0.0267	0.0150	0.0283	0.0152
16	0.0273	0.0155	0.0266	0.0149	0.0276	0.0154

Table 9: Span-coalescence width and sampling-method ablation on Amazon-Beauty.

D.3 Loss Component Ablation

Table 10 ablates the length-regularization weight λ_{len} and entropy coefficient β_{ent}. Removing either regularizer degrades performance, suggesting that they play complementary roles: entropy maintains exploration, while length regularization stabilizes expected selection density.

D.4 End-to-end Latency Breakdown

Table 11 reports wall-clock Actor latency across model sizes. Even the 4B Actor used in our most demanding setting adds under 0.34 s (p95), compared with 8–18 s for the 14B Solver. The total inference-time overhead of the Actor is therefore roughly 1.3%–2.9% of the pipeline latency. Our latency measurements are for *per-query* inference, where each query induces one Actor pass and one Solver call. Conversational multi-turn serving with cache-aware reuse across changing query-conditioned highlights would require an additional systems layer and is left to future work.

λ_{len}	β_{ent}	HR@10	NDCG@10
0.0	1.0	0.0267	0.0149
0.001	1.0	0.0271	0.0153
0.005	1.0	0.0284	0.0152
0.01	1.0	0.0287	0.0157
0.01	0.5	0.0276	0.0153
0.01	0.1	0.0278	0.0156
0.01	0.0	0.0268	0.0149

Table 10: Loss-coefficient ablation on Amazon-Beauty.

Latency (s)	**Actor 0.6B**	**Actor 1.7B**	**Actor 4B**	**Actor 8B**	**Solver 14B**
mean	0.047	0.102	0.233	0.528	8.077 to 18.403
p50	0.051	0.123	0.237	0.546	8.086 to 19.155
p95	0.057	0.130	0.336	0.635	9.085 to 19.589

Table 11: Actor latency across model sizes. Solver latency varies by task.

D.5 Sensitivity Analysis

Sensitivity to highlight budget. We vary the highlight budget γ, which caps the fraction of tokens the Actor may emphasize, and report performance on Amazon-Beauty in Figure 4. Two trends emerge. First, HILIGHT delivers gains even at small budgets (e.g., $\gamma \leq 0.10$), suggesting that emphasizing a small number of spans can already improve ranking in long histories. Second, performance is broadly stable over a wide mid-range ($\gamma \in [0.10, 0.30]$), with a mild optimum around $\gamma \approx 0.25$. This plateau indicates that HILIGHT is not brittle to γ: once the budget is sufficient to cover the key spans, additional highlighting yields diminishing returns and may slightly reduce contrast between emphasized evidence and distractors. *Practically, this means* HILIGHT *can be deployed without extensive per-task budget tuning.*

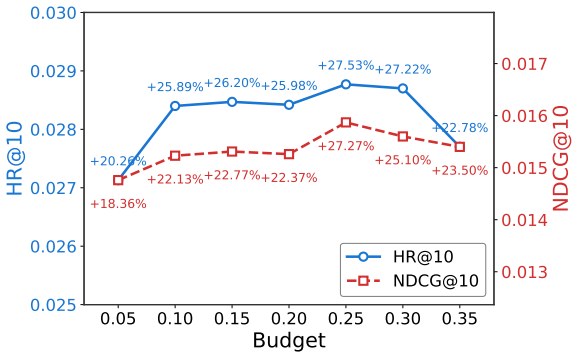


Figure 4: Sensitivity to highlight budget γ on Amazon-Beauty. Percentages denote improvement over MI.

Sensitivity to highlight markers. We test whether HILIGHT 's gains depend on the specific tag tokenization used to denote emphasis. Concretely, we *train a single Actor using the default marker format* (the one used in Table 1) and, at inference time, *swap the marker syntax* while keeping the selected spans, checkpoint, context, decoding settings, and highlight budget fixed. This isolates marker-tokenization effects from span-selection quality, since the Actor and selected spans are unchanged. The emphasized text content is identical; only the boundary tokens that denote emphasis are replaced. Table 12 shows that performance varies only modestly across Markdown-style, symbol-delimited, and HTML/XML-style markers: while the default format remains best, alternative syntaxes retain most of the gain. This indicates the learned policy is not overfitting to a particular delimiter tokenization.

Sensitivity to Solver prompt templates. We further vary the Solver prompt on HotpotQA using five templates: the default instruction used in the paper; two independent paraphrases with identical

Variant	HR@10	NDCG@10
`**...**`	0.02837	0.01512
`[[...]]`	0.02852	0.01547
`{...}`	0.02852	0.01547
`»...«`	0.02853	0.01512
`<b>...</b>`	0.02852	0.01575
`<important>`	0.02854	0.01570
Default	0.02877	0.01587

Table 12: Marker-format robustness on Amazon-Beauty. We train HILIGHT with the **default** marker format (Table 1) and replace *only* boundary-marker tokens at inference time (selected spans unchanged). Performance is stable across Markdown, symbol, and HTML/XML-style syntaxes.

content, the second in a more verbose register; a minimal template that keeps only the bare answer-format contract; and an order-swapped template that uses the default instruction but places the evidence before the question. Table 13 shows that HILIGHT yields higher EM and F1 under all five templates, with relative improvements of 4.23% to 9.29% in EM and 3.61% to 7.67% in F1. The minimal template lowers absolute performance for both conditions, suggesting that prompt quality and evidence emphasis provide complementary benefits.

	MI		HILIGHT	
Template	**EM**	**F1**	**EM**	**F1**
Default	0.556	0.706	0.606 (+8.99%)	0.741 (+4.96%)
Paraphrase A	0.574	0.711	0.606 (+5.57%)	0.749 (+5.34%)
Paraphrase B	0.554	0.690	0.592 (+6.86%)	0.736 (+6.67%)
Order swapped	0.560	0.704	0.612 (+9.29%)	0.758 (+7.67%)
Minimal	0.426	0.581	0.444 (+4.23%)	0.602 (+3.61%)

Table 13: Solver prompt-template sensitivity on HotpotQA. Percentages are relative improvements of HILIGHT over MI under the same template.

Sensitivity to Actor capacity. We fix the Solver at Qwen3-14B and vary Actor size from 0.6B to 8B. Table 14 shows that Actor scaling helps most on harder evidence-selection tasks (HotpotQA), while gains are smaller on tasks where evidence is more localized (Amazon-Beauty). Returns diminish beyond mid-sized Actors (around 4B). In practice, this means that Actor sizing can be guided by the structural complexity of the evidence-selection problem rather than the raw context length.

Actor	Amazon-Beauty		HotpotQA	
	HR@10	NDCG@10	EM	F1
0.6B	0.02830	0.01521	0.593	0.721
1.7B	0.02877	0.01587	0.597	0.728
4B	0.02937	0.01620	0.606	0.741
8B	0.02994	0.01648	0.609	0.742

Table 14: Actor size vs. performance. Solver = Qwen3-14B.

Sensitivity to context length. To test HILIGHT under genuinely long contexts, we inject distractor documents into HotpotQA to extend input length from the original setting to approximately 8K, 16K, and 32K tokens. Table 15 shows that HILIGHT maintains its advantage even as context length and distractor load increase. While absolute gains decrease slightly at longer contexts, the relative improvement remains positive throughout, consistent with our hypothesis that HILIGHT targets evidence sparsity under distractors rather than absolute context length alone.

Evidence selection and Actor cost versus context length. To examine how the Actor's evidence selection scales with length, we extend the first 300 examples of the HotpotQA validation split to approximately 8K, 16K, and 32K Actor tokens in two ways, holding fixed the 4B Actor, the frozen Qwen3-14B Solver, the document-order shuffling procedure, and a selection budget of $k = 4$

Method	Context Length			
	∼2K (original)	∼8K	∼16K	∼32K
MI (EM)	0.556	0.524	0.518	0.512
HILIGHT (EM)	0.606	0.550	0.542	0.532
$\Delta_{\%}$	+9.0%	+5.0%	+4.6%	+3.9%
MI (F1)	0.706	0.683	0.675	0.656
HILIGHT (F1)	0.741	0.710	0.693	0.682
$\Delta_{\%}$	+5.0%	+4.0%	+2.7%	+4.0%

Table 15: HotpotQA performance at increasing context lengths. Distractor documents injected to extend context.

sentences. *Injected distractors* append paragraphs drawn from other validation questions as additional documents. *Wikipedia extensions* retain every original HotpotQA paragraph verbatim and extend it with the continuation of its Wikipedia article, removing the introduction to avoid duplication and dropping reference-style sections; if the source articles are exhausted before the target length, additional complete Wikipedia articles are appended as standalone documents. Retaining the original paragraphs verbatim keeps the annotated supporting sentences traceable. Supporting-fact recall@k is the fraction of annotated supporting sentences covered by the top four selected sentences; zero-gold failure is the fraction of examples for which no selected sentence is annotated as supporting evidence; and both-hop coverage is the fraction of examples whose selection contains at least one supporting sentence from each of the two gold paragraphs.

Table 16 shows that Wikipedia extensions yield lower supporting-fact recall and both-hop coverage than injected distractors at all three lengths. From 8K to 32K tokens, recall decreases from 0.755 to 0.689 with injected distractors and from 0.706 to 0.666 with Wikipedia extensions. At equal target lengths, the two settings have the same Actor resource profile: mean Actor latency is 0.41, 0.95, and 2.35 seconds at 8K, 16K, and 32K tokens, compared with 0.23 seconds at the original length of about 2K tokens, and peak incremental GPU memory above the post-loading baseline is 4.4, 8.5, and 17.2 GB, compared with 0.8 GB. All measurements were obtained with the Actor and Solver resident on a single 96 GB GPU, and no out-of-memory failures occurred.

Context	Extension	Recall@k	Zero-gold failure	Both-hop coverage
8K	Injected distractors	0.755	1.3%	0.593
8K	Wikipedia extensions	0.706	5.0%	0.537
16K	Injected distractors	0.739	1.3%	0.577
16K	Wikipedia extensions	0.681	6.3%	0.490
32K	Injected distractors	0.689	5.7%	0.507
32K	Wikipedia extensions	0.666	5.7%	0.470

Table 16: Evidence selection of the 4B Actor on 300 HotpotQA validation examples extended to increasing Actor-token lengths, with a selection budget of $k = 4$ sentences.

D.6 Transfer to API-Based Solvers

To validate practical black-box compatibility beyond open-weight models, we apply the same Actor trained with Qwen3-14B zero-shot to GPT-5 mini via its API. Table 17 shows that HILIGHT yields positive gains on both Amazon-Beauty and HotpotQA. The gains are more modest than in high-distractor recommendation settings, which is expected because the target Solver is substantially stronger and may already resolve more of the context on its own. Nevertheless, the result is important: HILIGHT can be deployed with an API-only Solver at inference time, without Solver gradients, logits, or internal activations.

Additional API-based Solvers. We further apply the same Actor without adaptation to five additional API-based Solver configurations on HotpotQA: GPT-5 (low), GPT-5 (high), Claude Haiku 4.5, Claude Sonnet 5, and Claude Opus 4.8. Table 6 in Section 4.6 reports relative improvements over each target Solver's own MI baseline, together with GPT-5 mini from Table 17. Both EM and F1 improve for all

Solver	Amazon-Beauty		HotpotQA	
	HR@10	NDCG@10	EM	F1
GPT-5 mini (MI)	0.03083	0.01941	0.646	0.809
GPT-5 mini + HILIGHT	0.03127	0.02070	0.657	0.818
$\Delta_{\%}$	+1.43%	+6.65%	+1.70%	+1.11%

Table 17: GPT-5 mini evaluation. Actor trained on Qwen3-14B and applied zero-shot.

six configurations. Within each family, the gains are smaller for stronger Solvers, consistent with stronger Solvers already resolving more of the context on their own; confirming this trend statistically would require uncertainty estimates for the paired differences.

D.7 Statistic Confidence

To assess the statistical reliability of the observed improvements, we report 95% confidence intervals computed via a paired bootstrap procedure. For each comparison between HiLight and a baseline B, we draw $R = 1000$ bootstrap replicates. In each replicate, we sample $n = 1000$ example indices uniformly with replacement from the test set and apply the *same* sampled indices to both methods before computing the paired relative improvement

$$\Delta_b^{\%} = 100 \cdot \frac{\mathrm{Metric}_{\text{HiLight}}(S_b) - \mathrm{Metric}_B(S_b)}{\mathrm{Metric}_B(S_b)},$$

where S_b is the sampled subset. The 95% confidence interval is reported as the percentile interval

$$\left[\mathrm{percentile}_{2.5}\{\Delta_b^{\%}\}_{b=1}^{R},\ \mathrm{percentile}_{97.5}\{\Delta_b^{\%}\}_{b=1}^{R}\right].$$

Pairing is essential here: because both methods are evaluated on the same examples, jointly resampling indices removes the variance attributable to example difficulty and isolates the method-level effect, yielding tighter and more appropriate intervals than independently bootstrapping each method. We apply this procedure to all reported metrics (HR@10, NDCG@10, EM, and token-F1).

Dataset	Metric	$\Delta^{\%}$ (HILIGHT vs. Best Base.)	$\Delta^{\%}$ (HILIGHT vs. MI)
Amazon-Beauty	HR@10	$+4.63\%$ [$+0.59, +8.83$]	$+27.03\%$ [$+20.07, +34.47$]
	NDCG@10	$+10.74\%$ [$+6.42, +14.02$]	$+26.00\%$ [$+19.20, +33.44$]
HotpotQA	EM	$+2.12\%$ [$+0.60, +4.05$]	$+8.69\%$ [$+5.71, +10.78$]
	F1	$+2.58\%$ [$+1.02, +4.17$]	$+4.97\%$ [$+2.13, +6.89$]
SQuAD 2.0	EM	$+2.39\%$ [$+1.40, +4.08$]	$+12.62\%$ [$+8.54, +14.65$]
	F1	$+4.46\%$ [$+2.87, +6.14$]	$+10.95\%$ [$+7.02, +14.05$]
PubMedQA	Acc.	$+1.27\%$ [$+0.36, +2.52$]	$+1.87\%$ [$+1.07, +2.75$]
	M-F1	$+5.42\%$ [$+3.78, +6.97$]	$+5.42\%$ [$+3.78, +6.97$]

Table 18: Relative improvement $\Delta^{\%}$ of HILIGHT over the strongest baseline and over MI. Intervals that exclude 0 indicate statistically significant improvement at the 95% level.

D.8 Interpretability & Visualization

Interpretability. Figure 5 illustrates a representative Amazon-Beauty case study where HILIGHT effectively steers the frozen Solver through high-noise user history. While the baseline model is distracted by irrelevant interactions and ranks the ground truth item 14th , our Actor successfully isolates decisive signals by highlighting explicit feature alignment (e.g., a user's desire for "grips") and strong sentiment. These structural markers create a semantic bridge to the target product, improving its rank to 5th. This example underscores the *auditability* of our framework: unlike black-box soft attention, the highlighted spans provide a human-readable rationale (e.g., the matching of "grips") that explains *why* the model improved its prediction.

Visualization of actor scores. To better understand the Actor's selection behavior beyond discrete highlight tags, we visualize token-level importance scores on a representative HotpotQA instance in

Model	Metric	$\Delta^{\%}$ (HiLight vs. MI)
Qwen3-4B	HR@10	$+13.87\%$ [$+7.89, +20.63$]
	NDCG@10	$+7.93\%$ [$+3.41, +13.07$]
Qwen3-8B	HR@10	$+13.34\%$ [$+6.73, +19.21$]
	NDCG@10	$+12.26\%$ [$+5.93, +18.08$]
Gemma-3-27B	HR@10	$+15.74\%$ [$+9.21, +23.46$]
	NDCG@10	$+29.87\%$ [$+20.84, +40.63$]
Qwen3-32B	HR@10	$+7.83\%$ [$+1.97, +13.52$]
	NDCG@10	$+19.24\%$ [$+10.83, +30.38$]
Llama-3-70B	HR@10	$+18.43\%$ [$+10.67, +25.89$]
	NDCG@10	$+26.13\%$ [$+16.97, +34.84$]

Table 19: Transfer of HiLight's emphasis actor (trained on Qwen3-14B) to smaller and larger solver LLMs on Amazon-Beauty.

Dataset	Metric	$\Delta^{\%}$ (HiLight vs. MI)
Amazon-Beauty	HR@10	$+1.31\%$ [$+0.19, +2.87$]
	NDCG@10	$+6.14\%$ [$+0.83, +10.97$]
HotpotQA	EM	$+1.41\%$ [$+0.45, +2.13$]
	F1	$+1.04\%$ [$+0.18, +1.79$]

Table 20: Zero-shot transfer of HiLight's emphasis actor (trained on Qwen3-14B) to GPT-5 mini as the solver .

Figure 2. We observe a small number of pronounced peaks: the first aligns with the query tokens, suggesting the Actor conditions on the question before scoring the context, and the remaining peaks concentrate on a few context regions that substantially overlap with the annotated supporting evidence (red spans). Between these peaks, scores remain low, indicating that the Actor focuses its budget on a limited set of candidate evidence spans rather than broadly reweighting the full input. While this visualization is qualitative, the peaked structure is consistent with HiLight 's intended role as a budgeted evidence selector for multi-hop reasoning.

Quantitative evidence alignment. To move beyond qualitative examples, we quantify the overlap between Actor highlights and ground-truth supporting facts annotated in HotpotQA. We threshold Actor scores at 0.5 to obtain predicted evidence tokens and compute precision, recall, and F1 against the annotations. Table 7 shows that the Actor achieves up to 0.78 F1 overlap with human evidence, despite never seeing evidence labels during training. This suggests that optimizing for task reward alone can implicitly recover meaningful evidence structure, providing a form of weak supervision for evidence selection without requiring explicit annotations. Comparing with the Actor-size study (Table 14), the monotonic improvement in evidence F1 from 0.68 (0.6B) to 0.78 (8B) mirrors the downstream task gains, confirming that additional Actor capacity translates into genuinely better evidence identification rather than exploiting spurious patterns.

Failure analysis. We analyze cases where the Solver is correct under MI but becomes incorrect after applying HiLight. Table 21 shows the breakdown. For Amazon-Beauty, we randomly sample 1000 test points whose ground truth appears in the candidate set (examples without ground truth are filtered out); for HotpotQA, we sample 1000 points from its 7405-example test set. On HotpotQA, where annotated supporting facts enable fine-grained diagnosis, 16.5% of failures are due to misleading emphasis on non-decisive spans, while 33.8% occur when the Actor surfaces relevant evidence but the Solver still fails at downstream reasoning. Across both tasks, the number of newly correct predictions (success cases) substantially exceeds the number of newly incorrect ones (failure cases).

D.9 Matched Evidence-Selection Controls

Table 22 complements the fixed-Actor Pruned ablation in Table 2 with additional HotpotQA controls. All controls use the same frozen Qwen3-14B Solver and evaluation set as HiLight. The trained

Case Study: Amazon-Beauty Item Re-ranking

Query:
Given a user's history summary and a list of candidate items, re-rank the candidates according to their likelihood of being the next item of interest.

Context:
...title="Great brush cleaner", text="Great brush cleaner! Easy to use **but it would be better if it had grips on the back** so it doesn't move around the sink"
...title="I am in LOVE with these", text=" **I am in LOVE with these! I would buy again.** They're easy to pack because ..."

Comparison & Outcome:
- Standard Baseline Output: Ranked **14th** (Failed to retrieve)
- Ours Output: Ranked **5th** (Successfully improved ranking)
- Ground Truth Item y^*: e.l.f. Tone Adjusting Face Primer ... **Grips Makeup To Last**, Veg

Figure 5: A case study from the Amazon-Beauty dataset, with **purple** text emphasizing the evidence. HILIGHT successfully identifies the user's specific preference for "grips", allowing the frozen Solver to match it with the target product characteristic.

Failure Mode	Amazon-Beauty	HotpotQA
Misleading emphasis on non-decisive spans	N/A	16.5%
Evidence surfaced, but Solver still fails	N/A	33.8%
Total success (MI incorrect → HILIGHT correct)	168	63
Total failure (MI correct → HILIGHT incorrect)	28	20
Total test samples	1000	1000

Table 21: Failure-case breakdown for instances where HILIGHT changes a correct MI prediction into an incorrect one.

pruning selector and the trained LLM highlighter also use the same training prefix, downstream $0.5\,\mathrm{EM} + 0.5\,\mathrm{F1}$ reward, 40K Solver-query budget, and checkpoint-selection rule, thereby matching data access, supervision, shared Solver cost, and model selection.

Trained pruning selector. This control uses the same Actor architecture and GRPO procedure as HILIGHT, but places the pruning operator inside the RL loop, so that rewards are computed from the selected sentences alone.

Trained LLM highlighter. An LLM matching the HILIGHT Actor in size generates the IDs of sentences to highlight and is optimized with sequence-level GRPO ($G = 4$, with group-normalized advantages) using the same downstream reward. Its selections are rendered with HILIGHT's marker operator. Its full-context prefill and autoregressive decoding give a different Actor-side cost profile from HILIGHT's single token-scoring pass, while the supervision and the Solver-query budget remain matched.

Prompt-tuned Solver. We apply PEFT prompt tuning to the frozen Solver (bf16) by prepending 32 virtual tokens (≈164K trainable parameters) whose embeddings are initialized from a system prompt. We train with AdamW (learning rate 5×10^{-3}, batch size 8) for one epoch on 10,000 HotpotQA examples, applying cross-entropy loss only to the gold-answer target, and evaluate with greedy decoding on the HotpotQA distractor validation split, which is disjoint from the prompt-tuning examples. Unlike HILIGHT, this diagnostic requires gradient access to the Solver.

Off-the-shelf reranker. `bge-reranker-v2-m3` scores every question and sentence pair, and the top-k sentences are passed to the Solver without HotpotQA-specific adaptation.

GPT-5 mini API highlighter. GPT-5 mini selects up to k sentence IDs from the numbered context in a zero-shot setting. The selected sentences are rendered with the same marker operator, and the same frozen Qwen3-14B Solver produces the answer. This control requires one proprietary API call per example.

Method	EM	F1
MI	0.556	0.706
Prompt-tuned Solver	0.587	0.727
Trained pruning selector	0.563	0.707
Trained LLM highlighter	0.572	0.712
Off-the-shelf reranker	0.568	0.710
GPT-5 mini API highlighter	0.612	0.766
HILIGHT	0.606	0.741

Table 22: HotpotQA comparison with additional evidence-selection controls. All methods use the same frozen Qwen3-14B Solver and evaluation set; the trained pruning selector, the trained LLM highlighter, and HILIGHT share the optimization pool, downstream reward, 40K Solver-query budget, and checkpoint-selection rule.

D.10 Cross-Task Transfer

To test whether an Actor trained on one task remains useful on a related unseen task, we apply the HotpotQA Actor unchanged to SQuAD 2.0, without target-task fine-tuning or model selection. Table 23 reports changes relative to the MI baseline of the evaluation task under its native evaluation protocol. The transferred Actor improves EM by +3.58% and F1 by +4.64%, below the +12.80% and +11.44% of the task-specific SQuAD 2.0 Actor, indicating that an Actor trained on one QA task retains useful evidence-selection behavior on a related task.

Actor training task	Evaluation task	Setting	$\Delta_{\%}$EM	$\Delta_{\%}$F1
SQuAD 2.0	SQuAD 2.0	in-task reference	+12.80%	+11.44%
HotpotQA	SQuAD 2.0	zero-shot cross-task	+3.58%	+4.64%

Table 23: Cross-task transfer to SQuAD 2.0 ($\Delta_{\%}$ over the SQuAD 2.0 MI baseline).

E Additional Case Studies

We provide additional qualitative case studies on HotpotQA and PubMedQA to illustrate how evidence emphasis mitigates common long-context failure modes, such as entity-level confusion and evidence dilution, thereby improving the frozen Solver's final prediction.

Case Study: HotpotQA Multi-hop QA

Query:
According to the 2001 census, what was the population of the city in which Kirton End is located?

Context:
...Boston is a town and small port in Lincolnshire, on the east coast of England...
...The town itself had **a population of 35,124 at the 2001 census**...
...**Kirton End is a hamlet in the civil parish of Kirton in the Boston district of Lincolnshire, England**...

Comparison & Outcome:
- Standard Baseline Output: **341** (Incorrect; confused with the hamlet/village population)
- Ours Output: **35,124** (Correct)
- Ground Truth y^*: **35,124**

Figure 6: A case study from the HotpotQA dataset, with **purple** text emphasizing the decisive evidence. HILIGHT highlights (i) that Kirton End is in the Boston district and (ii) Boston's 2001 census population, enabling the frozen Solver to answer correctly.

Case Study: PubMedQA Biomedical Yes/No QA

Query:
Do wastewater treatment plants release large amounts of extended-spectrum β-lactamase-producing *Escherichia coli* into the environment?

Context:
...The determinants of the spread of extended-spectrum β-lactamase-producing *E. coli* (ESBLEC) in the community remain unclear...
...**we analyzed the ESBLEC population throughout an urban wastewater network**...
...**The *E. coli* load was higher in urban wastewater than in hospital wastewater**...

Comparison & Outcome:
- Standard Baseline Output: **No** (Incorrect)
- Ours Output: **Yes** (Correct)
- Ground Truth y^*: **Yes**

Figure 7: A case study from the PubMedQA dataset, with **purple** text emphasizing the key evidence. By highlighting the study's findings on ESBLEC prevalence and *E. coli* load in wastewater, HILIGHT helps the frozen Solver infer the correct "Yes" answer.

F Reproducibility

F.1 Actor Model Architecture

Backbone LLM. We implement the actor on top of a pretrained causal language model (Qwen family). Given an input sequence $\{x_i\}_{i=1}^{L}$ (token IDs) and an attention mask, we run a standard forward pass to obtain per-layer hidden representations $\{\mathbf{h}_i^{(\ell)}\}$. To reduce overhead and keep the actor purely as a feature extractor, we disable KV caching.

Freezing the backbone. During actor training, we freeze the pretrained backbone parameters and prevent gradient updates to the full LLM. Concretely, we set $\nabla_{\theta_{\text{LM}}} = 0$, so the LLM serves as a fixed encoder. This substantially reduces the number of trainable parameters and improves training stability.

LoRA-based parameter-efficient adaptation. To enable task-specific adaptation without updating the full backbone, we optionally attach LoRA adapters. We adopt a standard LoRA configuration with rank $r = 8$, scaling factor $\alpha = 16$, and dropout 0.05, and inject adapters into the self-attention projection layers: $\{\texttt{q_proj}, \texttt{k_proj}, \texttt{v_proj}, \texttt{o_proj}\}$. With the backbone frozen, only the LoRA parameters and the lightweight actor heads are optimized, yielding an efficient training setup.

Layer mixing for token representations. Instead of relying on a single transformer layer, we form token features using a learned convex combination of the last four hidden layers. Let $\mathbf{h}_i^{(-4)}, \ldots, \mathbf{h}_i^{(-1)}$ denote the last four layer outputs. We learn a parameter vector $\mathbf{a} \in \mathbb{R}^4$ and compute weights $w_k = \mathrm{softmax}(\mathbf{a})_k$. The mixed token representation is:

$$\mathbf{x}_i = \sum_{k=1}^{4} w_k \, \mathbf{h}_i^{(-k)}. \tag{11}$$

This aggregation often provides more robust saliency features than using only the final layer.

Policy and value heads. On top of $\mathbf{x}_i$, we attach a lightweight linear head with a LayerNorm. The *policy head* produces token-level logits z_i and converts them to Bernoulli importance probabilities via a temperature-controlled sigmoid:

$$p_i = \sigma\left(\frac{z_i}{\tau}\right), \tag{12}$$

where τ is the temperature and $i \in \{1, \ldots, L\}$ indexes token positions. These probabilities parameterize the actor's token-wise importance policy: p_i is the probability of marking token i as important.

Overall, our actor combines a frozen pretrained LLM backbone with parameter-efficient LoRA adapters and a small policy head, yielding a compact and stable model for token-level emphasis learning.

F.2 Data Preprocessing

Amazon-Beauty. For each user, we parse the chronological interaction sequence (purchase history) as a list of item IDs $\mathcal{H}_u = [item_1, item_2, \ldots, item_T]$. We formulate next-item recommendation by using the *last purchased item* as the supervision label, $y_u = i_T$, and the remaining prefix as the model input history, $\mathcal{H}'_u = [item_1, \ldots, item_{T-1}]$. Users with fewer than two interactions are discarded since they do not provide a valid (history, label) pair.

We build an item-item co-visitation (co-vis) graph from training histories using a sliding-window scheme. For each user history $\mathcal{H}_u$, we iterate over positions t and add weighted edges between item pairs that co-occur within a window of radius v:

$$\forall\, j \in [t-v,\, t+v],\ j \neq t: \quad \mathrm{co}[item_t][item_j] \mathrel{+}= 1.$$

Aggregating over all users yields a directed adjacency map where each item a has a neighbor list $\mathcal{N}(a) = \{(b,\, \mathrm{co}[a][b])\}$ sorted by descending co-visitation count.

For each user history $\mathcal{H}'_u$, we construct a candidate set $\mathcal{C}_u$ by retrieving neighbors from the co-vis graph, emphasizing recent interactions. Let $\mathcal{R}_u$ be the last M items of the history (we use $M = 10$) in reverse chronological order. We score each candidate item $c \notin \mathcal{H}'_u$ by accumulating co-vis weights from recent items with an exponential recency decay:

$$s(c) = \sum_{r \in \mathcal{R}_u} \alpha^{\mathrm{rank}(r)} \cdot \mathrm{co}[r][c],$$

where $\alpha \in (0, 1)$ controls the decay (we use $\alpha = 0.85$) and $\mathrm{rank}(r) = 0$ denotes the most recent item. We then take the top-K items by score as co-vis candidates. If fewer than K candidates are available, we fill the remaining slots with the most popular items not appearing in the history nor already selected. By default, we select 40 candidates for both training and evaluation.

Other datasets. For other datasets, we use the default dataset loader from [2]Hugging Face.

F.3 Hyperparameter Summary

Table 24 summarizes the key hyperparameters across tasks; we additionally fix the training batch size to 1 and the sliding-window radius to 3 for candidate-set construction.

Symbol	Default	Description
τ	1.0	Temperature in Eq. (3) for scaling token-importance logits before the sigmoid
γ	0.25 (Beauty), 0.15 (others)	Highlight budget fraction; $k = \lfloor \gamma L \rfloor$ tokens selected after projection
δ	10	Gap-bridging threshold in span coalescence: spans separated by $\leq \delta$ tokens are merged
G	4	Number of sampled masks per instance in grouped policy-gradient training
Ω	All context tokens	Set of policy-controlled tokens (query tokens are excluded from highlighting)
λ_{len}	0.01	Weight for target-length regularization (Eq. (9))
β_{ent}	1.0	Entropy bonus coefficient (Eq. (8))
lr	1×10^{-4}	Adam learning rate
wd	1×10^{-2}	Weight decay
LoRA r	8	LoRA rank for Actor backbone adaptation
LoRA α	16	LoRA scaling factor

Table 24: Consolidated hyperparameter table.

[2]https://huggingface.co/docs/datasets/en/index

F.4 Training Settings

Amazon-Beauty. For the Amazon-Beauty recommendation task, the actor takes only the user's historical reviews as input. The candidate set (along with each item's metadata, e.g., title/brand/category/price band and short phrases) is treated as part of the query used by the downstream solver, rather than being fed into the actor. In this way, the actor focuses on selecting salient preference signals from the user history, while the re-ranker conditions on both the emphasized history and the candidate descriptions.

Question answering. For QA tasks, we feed the actor a concatenation of the question and the evidence, but we apply highlighting only to the evidence span. This design encourages the actor to identify the minimal supporting evidence for answering, while keeping the question intact as a conditioning prompt for the downstream QA model.

F.5 Training Protocol and Data Access

Table 25 lists the data source, training use, and evaluation set for each task. Actor training consumes a sequential prefix of each training source in a single pass. With batch size 1 and one RL update per training instance, Amazon-Beauty uses 3,000 unique training instances and 3,000 RL updates, while HotpotQA, SQuAD 2.0, and PubMedQA each use 10,000 unique training instances and 10,000 RL updates. Each update evaluates $G = 4$ sampled masks, resulting in 12K optimization-time Solver queries for Amazon-Beauty and 40K for each QA task. Checkpoints are saved every 1K updates. For each dataset, checkpoint selection uses a fixed 500-example subset sampled with seed 42 from that dataset's training split; these examples are disjoint from both the instances used for Actor optimization and all final evaluation examples. In particular, none of the 7,405 HotpotQA distractor-validation examples used for final evaluation is used for checkpoint selection.

Dataset and configuration	Training source and use	Evaluation source and use
Amazon-Beauty; final item for each user as the label; 40 co-visitation candidates	`train_split`: 3,470,973 rows; first 3,000 used for Actor training	`test_split`: 867,744 rows; users sampled (seed 42) until 1,000 have the ground truth among the 40 candidates; metrics over all 18,633 sampled user
HotpotQA; `hotpot_qa`, `distractor` configuration (Hugging Face)	90,447 rows; first 10,000 processed sequentially in one pass	Full distractor validation split: 7,405 examples (official test labels are hidden)
SQuAD 2.0; `squad_v2` (Hugging Face)	130,319 rows; first 10,000 processed sequentially in one pass	2,000 examples sampled from the validation split (seed 42)
PubMedQA; `pubmed_qa`, `pqa_artificial` configuration (Hugging Face)	211,269 rows; first 10,000 processed sequentially in one pass	Fixed evaluation sample of 1,000 examples

Table 25: Data splits and their use for Actor training and evaluation.

Table 26 reports the optimization-time Solver queries of all methods on the four tasks, and Table 27 summarizes their training-data access, supervision signals, and selection rules. All learned methods draw from the same per-task optimization pool. The prompt optimizers use larger Solver-query budgets and, depending on the method, observe gold demonstrations or meta-prompt feedback, whereas HILIGHT receives only the scalar downstream reward, without in-context gold outputs or token-level evidence labels. MI and SELF-MARK use no optimization data or optimization-time Solver queries, although SELF-MARK adds one extra Solver query per example at inference time.

F.6 Prompts

In this section, we provide the prompts used in our experiments.

Solver LLM Re-ranking Prompt for Amazon-Beauty (Training)

```
You are a recommender re-ranker. Your goal is to score each candidate from 0
↪  to 10 based on how likely it is to be the user's next item.
```

Method	Optimization or control	Instance-conditioned	Beauty	HotpotQA	SQuAD	PubMedQA
MI	Fixed manual instruction	No	0	0	0	0
PRL	Task-level learned prompt	No	120K	400K	400K	400K
BFRS	Prompt and demonstration search	No	60K	200K	200K	200K
DSPy/MIPROv2	Prompt-program optimization	No	24K	80K	80K	80K
APE	Instruction induction	No	60K	200K	200K	200K
OPRO	Instruction optimization	No	30K	100K	100K	100K
SELF-MARK	Zero-shot self-highlighting	Yes	0	0	0	0
HILIGHT	Learned context-span emphasis	Yes	12K	40K	40K	40K

Table 26: Optimization-time Solver queries for each method and task.

Method	Training-data access	Supervision signal	Selection basis
MI	none	none	fixed instruction
PRL	shared per-task subset	task-metric reward on candidate prompts; failure cases with gold answers shown in the meta-prompt	best instruction by subset score
BFRS	shared per-task subset	task metric; self-bootstrapped demonstrations (gold outputs of solved examples)	best-first search score on subset
DSPy/MIPROv2	shared per-task subset	task metric via surrogate-guided evaluation; grounded demonstrations with gold outputs	best configuration by surrogate or evaluation score
APE	shared per-task subset	gold input and output demonstrations; candidate scoring on subset	highest-scoring instruction
OPRO	shared per-task subset	scores of prior candidates and gold demonstrations in the meta-prompt	top instruction by subset score
SELF-MARK	none	none	not applicable (zero-shot)
HILIGHT	shared per-task subset	outcome-level task reward only; no gold answers in context and no token-level evidence labels	checkpoint selected on a held-out subset

Table 27: Training-data access, supervision, and selection rules of each method.

```
Return JSON only, for example:
<FINAL_JSON>
[{"id": 1,"score": 8.5}, {"id":2,"score":6.0}, {"id": 10, "score": 0.7},
↪ {"id": 5, "score": 0.6}, ...]
</FINAL_JSON>

[USER_HISTORY_COMPACT]
<tagged_evidence>

[CANDIDATES]
cid=1 | title="<title>" | brand="<brand>" | cat="<cat>" |
↪ price_band="<price_band>" | rating=<avg_rating>(<rating_count>) |
↪ <phrases>
cid=2 | title="<title>" | brand="<brand>" | cat="<cat>" |
↪ price_band="<price_band>" | rating=<avg_rating>(<rating_count>) |
↪ <phrases>
...
Example Final JSON Output:

Rules:
1. Output ONLY the JSON array, nothing before or after.

2. Keep the format strictly valid JSON.
```

Solver LLM Re-ranking Prompt for Amazon-Beauty (Testing)

```
Given a user's history summary and a list of candidate items, re-rank the
↪ candidates according to their likelihood of being the next item of
↪ interest to the user based on the user's historical interactions. Each
↪ entry should contain an "id" and a "score" reflecting the predicted
↪ relevance of the candidate to the user. Ensure that the JSON is
↪ well-formed and encapsulated within <FINAL_JSON> tags. Some parts of the
↪ history are wrapped in <start_important> ... <end_important>. Treat those
↪ spans as especially important signals about the user preferences.
```

```
Example Final JSON Output:

<FINAL_JSON>
[{"id": 1,"score": 8.5}, {"id":2,"score":6.0}, {"id": 10, "score": 0.7},
↪  {"id": 5, "score": 0.6}, ...]
</FINAL_JSON>

Please adhere to the provided structure and ensure the JSON output is
↪  strictly formatted as shown.

[USER_HISTORY_SUMMARY]
<user_summary>

[CANDIDATES]
id=1 | title="<title>" | brand="<brand>" | cat="<cat>" |
↪  price_band="<price_band>" | rating=<avg_rating>(<rating_count>) |
↪  <phrases>
id=2 | title="<title>" | brand="<brand>" | cat="<cat>" |
↪  price_band="<price_band>" | rating=<avg_rating>(<rating_count>) |
↪  <phrases>

Final JSON Output:
```

QA tasks Training-Time Prompt

```
You are a helpful, precise QA assistant.
Follow the format EXACTLY:
You MUST output ONLY the short answer phrase inside <answer>...</answer>.
No explanation, no extra words.
Some parts of the EVIDENCE are wrapped in <start_important> ...
↪  <end_important>.

QUESTION:
<question>

EVIDENCE:
<evidence>

OUTPUT:
```

QA tasks Testing-Time Prompt

```
You are a helpful, precise QA assistant.
Follow the format EXACTLY:
You MUST output ONLY the short answer phrase inside <answer>...</answer>.
No explanation, no extra words.
Some parts of the EVIDENCE are wrapped in <start_important> ...
↪  <end_important>.

QUESTION:
<question>

EVIDENCE:
<evidence>

OUTPUT:
```

OPRO Meta-Prompt for Amazon-Beauty

```
You are an expert prompt engineer optimizing an instruction for a recommender reranking task.
The instruction will be used in a 0-shot prompt that asks the model to output ONLY:
[{"id": 1, "score": 7.5}, {"id": 2, "score": 3.0}, ...]

The model sees:
- A user history summary (text describing past interactions)
- A list of candidate items with metadata (title, brand, category, price, rating, review phrases)

Your job: propose {n} new instruction variants that might improve HR@10 and NDCG@10.

[TOP INSTRUCTIONS SO FAR]
1. "{instruction_1}" → HR@10={hr1:.3f}, NDCG@10={ndcg1:.3f}
2. "{instruction_2}" → HR@10={hr2:.3f}, NDCG@10={ndcg2:.3f}
...

Generate {n} new diverse instructions. Output each on its own line, no numbering, no quotes.
```

OPRO Meta-Prompt for QA tasks

```
You are an expert prompt engineer.
Goal: write an INSTRUCTION for question answering.
The instruction will be placed BEFORE QUESTION and EVIDENCE.

Hard constraints:

Answer using ONLY the provided EVIDENCE.
Be concise.
Output ONLY: <FINAL_ANSWER>answer</FINAL_ANSWER> (no extra text, no markdown, no headings).

Here are example INPUT/OUTPUT pairs (demos). Your instruction should make the model produce outputs like these:

===== DEMO 1 INPUT =====
QUESTION:
<demo_1_question>

EVIDENCE:
<demo_1_evidence>

===== DEMO 1 OUTPUT =====
<FINAL_ANSWER>
<demo_1_answer>
</FINAL_ANSWER>
...

Best instructions so far (higher F1/EM is better):

--- BEST 1 ---
F1=<f1> EM=<em> empty=<empty> n=<n>
<best_instruction_1>

--- BEST 2 ---
...

Now generate <n_new> NEW, diverse instruction candidates.
Return ONLY valid JSON (no markdown) in this schema:
```

```
{"instructions": ["...", "...", ...]}
Do not include any other keys. Do not include explanations.
```

PRL Meta-Prompt for Amazon-Beauty

```
You are an expert prompt engineer.

Task: Optimize an instruction for a recommender system that re-ranks
↪  candidate items.

The instruction will be inserted into a prompt template that includes:
- User history summary
- Candidate items with metadata

The model must output a JSON array: [{"id": int, "score": float}, ...]

[CURRENT TOP INSTRUCTIONS]
{ranked_instructions_with_scores}

[FEEDBACK]
{optional_feedback_from_failed_cases}

Generate {n} new instruction variants that could improve Hit Rate and NDCG
↪  metrics.
Be creative but maintain the core task requirements.
```

PRL Meta-Prompt for QA tasks

```
You are an expert prompt optimizer.
Task: improve an INSTRUCTION for question answering.
The instruction will be placed BEFORE QUESTION and EVIDENCE.

Hard constraints for ANY instruction you propose:

The model must answer based on ONLY the provided EVIDENCE.

Output ONLY: <FINAL_ANSWER>answer</FINAL_ANSWER> (no extra text, no markdown,
↪  no headings).

Be concise (often 1-8 words).
Current instruction:
----- CURRENT_INSTRUCTION -----
<current_instr>
----- END_CURRENT_INSTRUCTION -----

Current score on training subset: metric(F1+0.1*EM)=<metric> F1=<f1> EM=<em>
↪  empty=<empty> n=<n>

Top instructions so far (for context):
[TOP 1] metric=<metric> F1=<f1> EM=<em>
<top_instruction_1>
[TOP 2] ...

Here are some failure cases of the CURRENT instruction. Improve the
↪  instruction to fix these:
===== FAILURE 1 =====
QUESTION:
<failure_1_question>

EVIDENCE:
<failure_1_evidence>
```

```
GOLD:
<failure_1_gold>

PRED:
<failure_1_pred>

F1=<f1> EM=<em>
===== FAILURE 2 =====
...
Now propose <n_new> improved, diverse instruction candidates.
Each candidate should be a SMALL EDIT of the current instruction (not a
↪  completely different format), but you may add 1-3 extra rules that
↪  improve answer extraction and concision.

Return ONLY valid JSON (no markdown) in this schema:
{"instructions": ["...", "...", ...]}
No other keys. No explanations.
```

DSPy Base-Instruction for Amazon-Beauty

```
You are a recommendation re-ranker.
You will be given two blocks:
[USER_HISTORY_SUMMARY] ...
[CANDIDATES] lines with cid=... and metadata

Score EACH candidate by likelihood of being the user's NEXT interaction.
Use a 0-10 scale (higher = more likely) with meaningful spread (avoid ties).

STRICT output format (no extra text):
<FINAL_JSON>
[{"id": cid_int, "score": number}, ...]
</FINAL_JSON>

Rules:

Include EVERY candidate exactly once; id must match the provided cid.
Use user history signals (brand/category/phrases) to guide scores.
```

DSPy Base-Instruction for QA tasks

```
Question answering task.
You will be given a QUESTION and EVIDENCE context.
Answer the question using ONLY the evidence. Be concise.

STRICT output format (no extra text):
<FINAL_ANSWER>
your answer here
</FINAL_ANSWER>
```

APE-style Instruction Induction for Amazon-Beauty

```
Goal: write an INSTRUCTION for a recommender re-ranking task.
This instruction will be placed BEFORE the blocks [USER_HISTORY_SUMMARY] and
↪  [CANDIDATES].

The model must output ONLY this exact wrapper and JSON:
<FINAL_JSON>
[{"id": cid_int, "score": number}, ...]
</FINAL_JSON>

Hard constraints for your instruction:
```

```
Enforce strict output formatting: ONLY <FINAL_JSON>...</FINAL_JSON> with a
↪  JSON array.

Include EVERY candidate exactly once; "id" is the provided cid integer;
↪  "score" is a number.

Encourage meaningful score spread (avoid ties) and use user history signals.

Keep the instruction concise (<= 120 words).

Here are example INPUT/OUTPUT pairs (demos). The instruction you write should
↪  make the model produce outputs like these:

===== DEMO 1 INPUT =====
<demo_1_input>
===== DEMO 1 OUTPUT =====
<demo_1_output>

===== DEMO 2 INPUT =====
<demo_2_input>
===== DEMO 2 OUTPUT =====
<demo_2_output>

...

Here are the best instructions found so far and their evaluation metrics
↪  (higher NDCG/HR is better):

--- BEST 1 ---
NDCG=<ndcg> HR=<hr> empty_parse=<empty_parse> n=<n>
<best_instruction_1>

--- BEST 2 ---
NDCG=<ndcg> HR=<hr> empty_parse=<empty_parse> n=<n>
<best_instruction_2>

...

Now generate <n_new> NEW, diverse instruction candidates.
Return ONLY valid JSON (no markdown) in this schema:
{"instructions": ["...", "...", ...]}
Do not include any other keys. Do not include explanations.
```

APE-style Instruction Induction for QA tasks

```
You are an expert prompt engineer (APE).
Goal: write an INSTRUCTION for question answering.
The instruction will be placed BEFORE QUESTION and EVIDENCE.

Hard constraints:

Answer using ONLY the provided EVIDENCE.

Output ONLY: <FINAL_ANSWER>answer</FINAL_ANSWER> (no extra text, no
↪  markdown).

Be concise; prefer a short span/phrase or "yes"/"no" when appropriate.

Examples (demos):

===== DEMO 1 INPUT =====
QUESTION:
```

```
<demo_1_question>

EVIDENCE:
<demo_1_evidence>

===== DEMO 1 OUTPUT =====
<FINAL_ANSWER>
<demo_1_answer>
</FINAL_ANSWER>

===== DEMO 2 INPUT =====
QUESTION:
<demo_2_question>

EVIDENCE:
<demo_2_evidence>

===== DEMO 2 OUTPUT =====
<FINAL_ANSWER>
<demo_2_answer>
</FINAL_ANSWER>

...

Best instructions so far (higher metric = F1 + 0.1*EM is better):

--- BEST 1 ---
metric=<metric> F1=<f1> EM=<em> empty=<empty> n=<n>
<best_instruction_1>

--- BEST 2 ---
metric=<metric> F1=<f1> EM=<em> empty=<empty> n=<n>
<best_instruction_2>

...

Now generate <n_new> NEW, diverse instruction candidates.
Return ONLY valid JSON (no markdown) in this schema:
{"instructions": ["...", "...", ...]}
No other keys. No explanations.
```

BFRS Prompt for Amazon-Beauty

```
You are a recommender re-ranker.
Goal: score each candidate 0-10 for next-item likelihood for this user.

Output format rules:

You MUST output ONLY the final JSON array inside
↪  <FINAL_JSON>...</FINAL_JSON>.

<FINAL_JSON> must be a single valid JSON array of objects with fields "id"
↪  and "score".

No extra text outside <FINAL_JSON>.

The "id" field must be an integer matching id below.

The "score" field must be a number (int or float).

Here are some solved examples:

EXAMPLE 1
```

```
[USER_HISTORY_SUMMARY]
<example_1_user_summary>

[CANDIDATES]
id=1 | title="<...>" | brand="<...>" | cat="<...>" | price_band="<...>" |
↪  rating=<...>(<...>) | <phrases>
id=2 | title="<...>" | brand="<...>" | cat="<...>" | price_band="<...>" |
↪  rating=<...>(<...>) | <phrases>
...

<FINAL_JSON>
<example_1_gold_json>
</FINAL_JSON>

EXAMPLE 2

[USER_HISTORY_SUMMARY]
<example_2_user_summary>

[CANDIDATES]
cid=1 | title="<...>" | brand="<...>" | cat="<...>" | price_band="<...>" |
↪  rating=<...>(<...>) | <phrases>
cid=2 | title="<...>" | brand="<...>" | cat="<...>" | price_band="<...>" |
↪  rating=<...>(<...>) | <phrases>

<FINAL_JSON>
<example_2_gold_json>
</FINAL_JSON>

Now solve the next case:

[USER_HISTORY_SUMMARY]
<user_summary>

[CANDIDATES]
cid=1 | title="<title>" | brand="<brand>" | cat="<cat>" |
↪  price_band="<price_band>" | rating=<avg_rating>(<rating_count>) |
↪  <phrases>
cid=2 | title="<title>" | brand="<brand>" | cat="<cat>" |
↪  price_band="<price_band>" | rating=<avg_rating>(<rating_count>) |
↪  <phrases>
...
Output:
```

BFRS Prompt for QA tasks

```
You are an expert prompt optimizer (BFRS).
Task: improve an INSTRUCTION for question answering.
The instruction will be placed BEFORE QUESTION and EVIDENCE.

Hard constraints for ANY instruction you propose:

The model must answer using ONLY the provided EVIDENCE.
Output ONLY: <FINAL_ANSWER>answer</FINAL_ANSWER> (no extra text, no markdown,
↪  no headings).
Be concise (often 1-8 words); prefer a short span/phrase or "yes"/"no" when
↪  appropriate.

Current best instruction:
----- CURRENT_INSTRUCTION -----
<current_instr>
----- END_CURRENT_INSTRUCTION -----
```

```
Current score on training subset: metric(F1+0.1*EM)=<metric> F1=<f1> EM=<em>
↪  empty=<empty> n=<n>

Top instructions so far (for context):
[TOP 1] metric=<metric> F1=<f1> EM=<em>
<top_instruction_1>

[TOP 2] metric=<metric> F1=<f1> EM=<em>
<top_instruction_2>

...

Here are some failure cases of the CURRENT instruction. Improve the
↪  instruction to fix these:

===== FAILURE 1 =====
QUESTION:
<failure_1_question>

EVIDENCE:
<failure_1_evidence>

GOLD:
<failure_1_gold>

PRED:
<failure_1_pred>

F1=<f1> EM=<em>

===== FAILURE 2 =====
...

BFRS generation rule:

Start from the CURRENT instruction and TOP instructions as ‘‘parents’’.
Create each new candidate by a SMALL EDIT (do not change the output tag
↪  format).
You may add 1-3 extra rules that improve answer extraction (shortest span,
↪  exact match, yes/no) and reduce empty/wrong-format outputs.
Ensure diversity across candidates (different phrasing/rules), while staying
↪  concise.

Now propose <n_new> improved, diverse instruction candidates.
Return ONLY valid JSON (no markdown) in this schema:
{"instructions": ["...", "...", ...]}
No other keys. No explanations.
```

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

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

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 5 includes a dedicated Limitations paragraph.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

Justification: The paper does not present formal theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper specifies tasks, datasets, metrics, baselines, model choices, rewards, hyperparameters, prompts, and compute resources in Section 4 and Appendices D–F.1.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: Code is available at `https://github.com/yh-sh/HiLight`. The submission uses public datasets and describes the implementation details needed for reproducibility.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Section 4 describes the experimental setup, and Appendix D gives rewards, hyperparameters, ablations, latency, and prompt templates.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The paper reports point estimates, ablations, robustness checks, failure analyses, and confidence intervals for the main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The experiments report the GPU configuration in Section 4 and Appendix D; Appendix D.4 also reports inference latency across Actor and Solver models.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The work uses public benchmark datasets and frozen LLM evaluation without crowdsourcing, human-subject experiments, or collection of private user data.

   Guidelines:

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

10. **Broader impacts**

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

    Answer:[Yes]

    Justification: The paper discusses both potential positive and negative societal impacts in its final section.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper does not release a new high-risk pretrained model, image generator, or scraped dataset.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: The paper cites all external datasets and model assets at first use, uses them under their respective public licenses or provider terms, and does not redistribute the original datasets, model weights, or API outputs as new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The released code is documented with a README that covers environment setup, data download, training, and checkpoint evaluation; the paper does not release a new dataset or benchmark.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject experiments, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [Yes]

    Justification: LLMs are central to the method: the paper describes the frozen Solver LLM, the LM-based Emphasis Actor, the reward-only training protocol, and the prompts used for recommendation and QA.

    Guidelines:

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