AREA: ADAPTIVE RELEVANCE EVIDENCE ALLOCATION FOR EFFICIENT LLM RERANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language model (LLM) rerankers provide strong relevance modeling but incur substantial inference cost because full-context inference propagates every input representation through all model layers. Existing efficient reranking methods typically rely on predefined compression configurations or externally selected operating points, despite substantial variation in where relevance evidence occurs and how much evidence different query–document pairs require. We formulate efficient reranking as *adaptive relevance evidence allocation* and introduce AREA, a pointwise reranker that learns both where full-resolution intermediate representations should be preserved and how much evidence each instance requires for deeper computation. AREA derives block-level localization supervision from the sensitivity of the relevance objective to compression, and instance-level budget supervision from counterfactual ranking feedback, without explicit token- or budget-level annotations. Under a fixed 50% retention ratio before ranking-guided optimization, AREA achieves 28.81 nDCG@10 on BRIGHT, compared with 27.72 for LTC. With adaptive budgeting, AREA reaches 30.21 nDCG@10 at an average retention ratio of 0.49, comparable to the separately optimized full-context variant at 30.10, while providing a $1.33\times$ controlled model-forward speedup. Consistent results on industrial search further demonstrate the effectiveness of task-derived localization and instance-specific budget allocation.

1 INTRODUCTION

Modern search systems typically follow a multi-stage ranking pipeline, in which an efficient retriever first identifies a small set of candidate documents and a more expressive reranker subsequently refines their ordering (Manning et al., 2008; Nogueira & Cho, 2019). With the growing use of large language models (LLMs) for relevance modeling, pointwise rerankers are particularly attractive for production search because they score each query–document pair independently, enabling parallel and scalable serving (Nogueira et al., 2020; Zhuang et al., 2023; Ma et al., 2024). However, propagating every document token through all model layers incurs substantial computational cost, especially for long candidate documents (Guan et al., 2022). In practice, relevance evidence is often concentrated in only a fraction of the document, while different query–document pairs may require different amounts of retained evidence (Xie et al., 2027; Lu et al., 2026). Full-context inference nevertheless allocates maximal computation to every instance, while existing efficient reranking methods typically rely on predefined compression configurations or externally selected operating points rather than adapting retention to individual query–document pairs (Zhuang et al., 2026b; Wang et al., 2026). We therefore formulate efficient pointwise reranking as **adaptive relevance evidence allocation**: determining where relevance-bearing representations should be preserved and how much evidence should continue through deeper computation for each query–document pair.

A fundamental challenge, however, is how to learn such adaptive allocation from conventional relevance supervision (Choi et al., 2024). Standard reranking data typically provide only document-level relevance labels, which characterize the overall relevance of a document to a query but offer no direct supervision for either localizing relevance-bearing representations or assigning instance-specific retention budgets (Zhuang et al., 2026a; Zhang et al., 2025; Liu et al., 2026; Su et al., 2025; Zhuang et al., 2024a). These decisions occur at different granularities: localization requires fine-grained importance signals over intermediate representations, whereas budgeting requires instance-level allocation signals (Goyal et al., 2020; Ye et al., 2021; Guan et al., 2022; Zhuang et al., 2026b). This

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

creates a supervision granularity gap between the document-level judgments available in standard ranking data and the block- and instance-level signals required for adaptive evidence allocation.

Our key insight is that both forms of supervision can be derived from relevance learning itself. The sensitivity of the reranking objective to intermediate representation compression provides fine-grained signals for evidence localization, while ranking feedback under alternative retention budgets provides instance-level signals of evidence sufficiency. Together, these task-derived signals bridge the supervision granularity gap without requiring explicit token- or budget-level annotations.

To instantiate this principle, we develop **AREA**, an efficient pointwise LLM reranker that performs adaptive relevance evidence allocation through intermediate-layer compression. A lightweight importance predictor identifies blocks to preserve at full resolution, while the remaining blocks are pooled before subsequent layers. An instance-level budget predictor determines how much intermediate evidence to retain from counterfactual ranking supervision. The resulting compression policy is integrated into the reranker while preserving independent pointwise inference.

Experiments across public and industrial benchmarks demonstrate that AREA provides a strong ranking effectiveness–efficiency trade-off, while controlled analyses validate both components of adaptive relevance evidence allocation. On BRIGHT (Su et al., 2025), AREA (Adaptive) achieves 30.21 nDCG@10 at an average retention ratio of 0.49, compared with 30.10 for the separately optimized full-context variant. Under a controlled HuggingFace implementation, AREA (Adaptive) achieves a $1.33\times$ model-forward speedup over the full-context variant. Under a controlled fixed 50% retention ratio before ranking-guided optimization, AREA's evidence localization achieves 28.81 nDCG@10, compared with 27.72 for LTC. Instance-specific budgeting further outperforms fixed, importance-based, and shuffled-budget alternatives at comparable retention, showing that the gain depends on matching computation budgets to individual query–document pairs. Consistent results on the industrial benchmark, BRIGHT-Long, and BEIR (Thakur et al., 2021) further support the effectiveness of adaptive evidence allocation across retrieval settings.

Our contributions are threefold:

Adaptive relevance evidence allocation. We formulate efficient pointwise LLM reranking as an adaptive evidence allocation problem that jointly determines *where* full-resolution representations should be preserved and *how much* evidence each query–document pair requires for deeper computation.

Task-derived multi-granularity supervision. We derive block-level localization supervision from compression sensitivity of the relevance objective and instance-level budget supervision from counterfactual ranking feedback, enabling adaptive allocation without explicit token- or budget-level annotations.

Effective and efficient pointwise reranking. AREA achieves a strong effectiveness–efficiency trade-off across public, long-document, and industrial benchmarks. Controlled experiments further show that task-aligned localization and instance-specific budgeting outperform fixed, heuristic, and shuffled allocation alternatives under comparable retention budgets.

2 Related Work

2.1 LLM-based Reranking

Large language models have broadened the design space of document reranking beyond conventional discriminative relevance models. Existing approaches formulate ranking through pointwise, pairwise, setwise, or listwise interactions, exposing different trade-offs between ranking effectiveness and inference efficiency (Sun et al., 2023; Qin et al., 2024; Zhuang et al., 2024b). More recent studies further incorporate explicit reasoning and ranking-oriented optimization, using reinforcement learning or reasoning-intensive supervision to improve relevance estimation on challenging queries (Zhuang et al., 2026a; Zhang et al., 2025; Liu et al., 2026; Cai et al., 2026).

These methods primarily use relevance supervision to improve ranking predictions or reasoning. AREA instead uses task-level relevance signals to supervise internal computation allocation, learning where intermediate representations should be preserved and how much computation each query–document pair requires.

2.2 Token Reduction and Adaptive Computation

Prior work on efficient Transformers has shown that computation need not be allocated uniformly across tokens or layers. Representative approaches progressively eliminate low-importance representations, learn variable layer-wise sequence lengths, or dynamically determine how deeply individual tokens are processed (Goyal et al., 2020; Kim & Cho, 2021; Ye et al., 2021). More recent elastic frameworks support multiple token-, layer-, or model-level operating points within a single model (Wang et al., 2026). Together, these studies establish that Transformer computation can be reduced or reconfigured without processing every representation through every layer.

Reranking-specific methods exploit the same opportunity in different ways. LTC applies adaptive pooling to intermediate representations, preserving early query–document interactions before shortening the sequence processed by subsequent layers (Zhuang et al., 2026b). CompRank reduces redundant ranking computation through segment-wise token compression and task-aligned scoring (Lu et al., 2026), while SERank performs query-aware sentence-level filtering before LLM reranking to increase evidence density (Xie et al., 2027). These approaches demonstrate that substantial portions of the input or intermediate sequence can be removed or compressed while preserving ranking effectiveness.

AREA instead focuses on the supervision of adaptive computation allocation. It derives block-level localization from the sensitivity of the relevance objective to compression, and instance-level budgets from counterfactual ranking sufficiency. This provides task-derived supervision for both where and how much intermediate computation should be retained.

3 Method

AREA introduces adaptive relevance evidence allocation into a pointwise LLM reranker through intermediate-layer compression. We first learn task-aligned evidence localization from the compression sensitivity of the reranking objective while keeping the reranker fixed. We then fix the localization module and learn instance-specific budgets from counterfactual ranking feedback while optimizing the reranker under the resulting compression configurations. At inference time, AREA requires only the standard query–document input and preserves independent pointwise scoring.

3.1 Problem Formulation

Given a query q, a retriever returns a candidate set $\mathcal{D}_q = \{d_1, \ldots, d_n\}$, and a pointwise reranker independently assigns each candidate a relevance score $s_i = f_\theta(q, d_i)$. We introduce adaptive relevance evidence allocation within this pointwise inference paradigm, implemented through intermediate-layer structured compression.

At a selected compression layer ℓ_c, let $H_i^{(\ell_c)} = [h_{i,1}^{(\ell_c)}, \ldots, h_{i,L_i}^{(\ell_c)}]$ denote the intermediate representations of (q, d_i). A compressible span is partitioned into contiguous blocks $\mathcal{B}_i = \{B_{i,1}, \ldots, B_{i,M_i}\}$. For each query–document pair, the model predicts block importance scores $\mathbf{a}_i = [a_{i,1}, \ldots, a_{i,M_i}]$ and an instance-specific retention budget b_i. Together, they define the compressed representations

$$\widetilde{H}_i^{(\ell_c)} = \mathcal{C}\Big(H_i^{(\ell_c)}; \mathbf{a}_i, b_i\Big),$$

where $\mathbf{a}_i$ determines *where* full-resolution representations are preserved and b_i determines *how much* of the intermediate sequence is retained for deeper computation. Neither signal is explicitly provided by standard relevance labels; AREA derives both from the reranking objective and ranking feedback while preserving independent pointwise inference.

3.2 Intermediate-Layer Structured Compression

The compression operator $\mathcal{C}$ is applied at an intermediate layer ℓ_c, after the preceding Transformer layers process the sequence at full resolution. The compressible span is partitioned into M contiguous blocks of size m. Each block is either *protected*, retaining all of its token representations, or *pooled* into a single mean representation. The resulting shortened sequence is then propagated through the remaining layers.

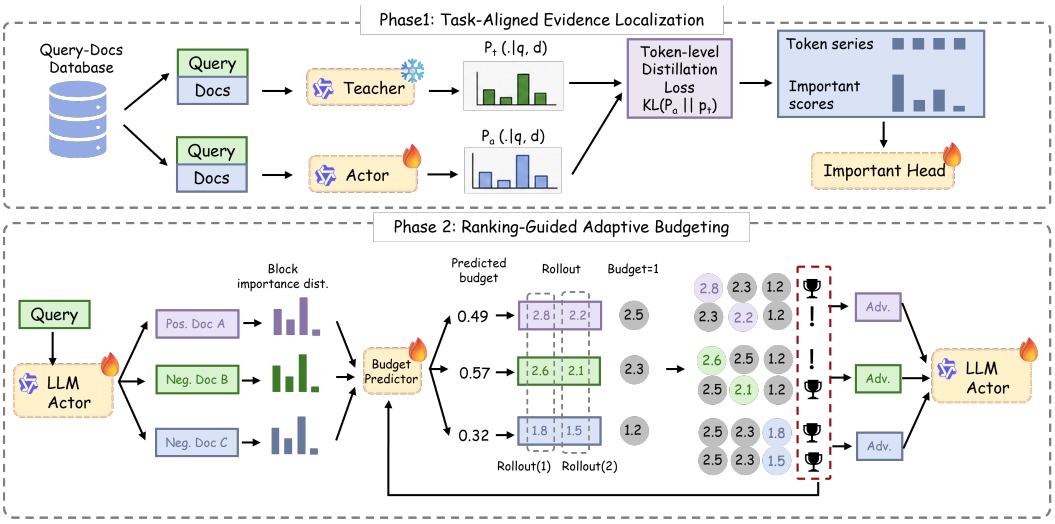


Figure 1: Overview of AREA. Task-aligned evidence localization learns where full-resolution representations should be preserved, while ranking-guided adaptive budgeting learns how much evidence each query–document pair should retain for deeper computation.

We define the retention budget b as the fraction of representations retained within the compressible span. For M equal-sized blocks with P protected blocks,

$$b \equiv \frac{L_{\text{comp}}}{L_{\text{orig}}} = \frac{M + P(m-1)}{mM}.$$

At execution, a predicted continuous budget is mapped to a feasible number of protected blocks under this relation. Given the resulting budget, blocks are ranked by their predicted importance, and the top-P blocks are protected while the remainder are pooled. The same Protect-or-Pool operator is used to define the localization supervision in Section 3.3.

3.3 Task-Aligned Evidence Localization

Relevance Distillation Objective. The student is the Qwen3.5-9B reranker after cold-start supervised fine-tuning. We construct a Qwen3.5-27B teacher using the same SFT procedure and keep it fixed during distillation. For each query–document pair (q, d), we apply on-policy distillation along trajectories τ generated by the student. Let $p_T^{(t)}$ and $p_S^{(t)}$ denote the teacher and student predictive distributions at step t, respectively. The distillation objective is

$$\mathcal{L}_{\text{dist}} = \sum_t D_{\text{KL}}\left(p_T^{(t)} \,\|\, p_S^{(t)}\right).$$

We use $\mathcal{L}_{\text{dist}}$ as the task objective for deriving compression-aware localization supervision.

Counterfactual Compression Importance. We define block importance by the sensitivity of the distillation objective to compressing that block. Let $\mathcal{L}_{\text{dist}}^{\text{full}}$ denote the loss with the full intermediate sequence and $\mathcal{L}_{\text{dist}}^{\text{pool}(B_j)}$ the loss after applying Protect-or-Pool to block B_j, replacing it with a single pooled representation and propagating the resulting shortened sequence through the remaining layers. The exact counterfactual importance is

$$I_j^* = \left|\mathcal{L}_{\text{dist}}^{\text{pool}(B_j)} - \mathcal{L}_{\text{dist}}^{\text{full}}\right|.$$

Evaluating I_j^* for every block requires a separate compressed forward pass. We therefore construct a same-length local surrogate in which each representation in B_j is replaced by the block mean $\bar{h}_j$ while the sequence structure is kept fixed. Let $g_k = \partial \mathcal{L}_{\text{dist}} / \partial h_k^{(\ell_c)}$. The first-order change under

this surrogate is

$$\widetilde{\Delta}_j \approx \sum_{k \in B_j} g_k^\top \left(\bar{h}_j - h_k^{(\ell_c)} \right).$$

We use its magnitude

$$I_j = \left| \sum_{k \in B_j} \left(h_k^{(\ell_c)} - \bar{h}_j \right)^\top g_k \right|$$

as an efficient proxy for I_j^* and the block-level localization target. Token-wise contributions are aggregated before taking the magnitude. Appendix B.3 empirically evaluates the fidelity of this first-order surrogate against the exact counterfactual importance computed with the actual Protect-or-Pool operator.

Importance Prediction. The resulting targets supervise a lightweight predictor g_ϕ at layer ℓ_c, which predicts token-level scores that are averaged within each block to obtain $\hat{s}_j$. We convert the compression sensitivities into a soft target distribution $\pi = \mathrm{softmax}(I)$ and train the predictor with cross-entropy against the softmax-normalized predicted block scores. The reranker remains fixed during this stage and only g_ϕ is updated. At inference time, block importance is predicted directly from the intermediate representations.

3.4 Ranking-Guided Adaptive Budgeting

Continuous Budget Prediction. For each query–document pair (q, d_i), let $\mathbf{a}_i = [a_{i,1}, \ldots, a_{i,M_i}]$ denote the block importance scores. We normalize them over the valid blocks, $x_{i,j} = \exp(a_{i,j}) / \sum_{r=1}^{M_i} \exp(a_{i,r})$, and right-pad the resulting sequence to a fixed length, yielding $\widetilde{\mathbf{x}}_i$. A lightweight predictor f_ψ outputs a continuous retention budget,

$$\hat{b}_i = b_{\min} + (b_{\max} - b_{\min}) \sigma(f_\psi(\widetilde{\mathbf{x}}_i)), \qquad \hat{b}_i \in [b_{\min}, b_{\max}].$$

The localization module remains fixed during budget learning, and $\hat{b}_i$ is mapped to a feasible Protect-or-Pool configuration as described in Section 3.2.

Directional Counterfactual Budget Supervision. For each query q, let $\mathbf{y} = [y_1, \ldots, y_n]$ denote the relevance labels and $\mathbf{z} = [z_1, \ldots, z_n]$ the ranking scores. Let $\mathcal{P} = \{i : y_i = \max_j y_j\}$ denote the positive documents and $\mathcal{N}$ the remaining candidates. For each $p \in \mathcal{P}$, we rank $\mathcal{G}_p = \{p\} \cup \mathcal{N}$ by predicted relevance and assign

$$R_p = \frac{1}{\mathrm{rank}_p(p)}, \qquad R_n^{(p)} = \begin{cases} -\dfrac{1}{\mathrm{rank}_p(p)}, & \mathrm{rank}_p(n) \le \mathrm{rank}_p(p), \\ 1 - \dfrac{(z_n - y_n)^2}{9}, & \mathrm{rank}_p(n) > \mathrm{rank}_p(p). \end{cases}$$

For multiple positives, negative rewards are averaged across groups; we denote the resulting per-document reward by $\mathcal{R}_i(\mathbf{z}, \mathbf{y})$.

To obtain instance-specific budget feedback, we vary the budget of one candidate at a time while keeping all other candidates at their full-context scores. Let

$$\mathbf{z}^{\mathrm{full}} = [z_1^{\mathrm{full}}, \ldots, z_n^{\mathrm{full}}], \qquad \mathbf{z}_i(b) = [z_1^{\mathrm{full}}, \ldots, z_i(b), \ldots, z_n^{\mathrm{full}}],$$

where

$$z_i(b) = \sum_d d \, p_i^{(b)}(d).$$

We define $R_i^{\mathrm{full}} = \mathcal{R}_i(\mathbf{z}^{\mathrm{full}}, \mathbf{y})$ and $R_i(b) = \mathcal{R}_i(\mathbf{z}_i(b), \mathbf{y})$. A budget b is considered candidate-wise sufficient under the fixed-full reference when

$$R_i(b) \ge R_i^{\mathrm{full}} - \tau.$$

Starting from the feasible configuration induced by $\hat{b}_i$, we search directionally over the discrete Protect-or-Pool budget lattice. If the current configuration is sufficient, we probe smaller budgets; otherwise, we probe larger ones. The search begins with an increment corresponding approximately to 0.05 retention and expands when necessary, up to approximately 0.20. The smallest confirmed sufficient budget is used as the regression target; when only a valid upper bound is identified, it provides one-sided supervision for the budget predictor.

Listwise Ranking Optimization. For each document d_i, the main pass executes the feasible compression configuration induced by $\hat{b}_i$ and produces a relevance distribution p_i^{main}. For each of G rollouts, we combine the sampled relevance label $\tilde{y}_i^{(g)}$ with the expected prediction μ_i^{main} to obtain a continuous ranking score,

$$c_i^{(g)} = \frac{\tilde{y}_i^{(g)} + \mu_i^{\text{main}}}{2}, \qquad \mu_i^{\text{main}} = \sum_d d\, p_i^{\text{main}}(d).$$

Using the fixed-full construction above, we replace the score of d_i by $c_i^{(g)}$ and compute the rollout reward $r_i^{(g)} = \mathcal{R}_i(\mathbf{z}_{i,g}, \mathbf{y})$. For the G rollouts of the same document, the group-relative advantage is

$$A_i^{(g)} = \frac{r_i^{(g)} - \bar{r}_i}{\sigma_i + \epsilon}.$$

The reranker parameters θ are optimized under the executed compression configuration using

$$\mathcal{L}_{\text{actor}} = \mathcal{L}_{\text{GRPO}} + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}.$$

During this stage, the localization predictor g_ϕ remains fixed. The trainable reranker parameters θ are updated with the listwise RL objective, whereas the budget predictor f_ψ is optimized separately from counterfactual supervision and receives no policy-gradient updates.

4 Experiments

We conduct experiments to evaluate AREA from both system-level and mechanism-level perspectives. Specifically, we ask: (1) whether AREA provides a competitive effectiveness–efficiency trade-off compared with existing LLM rerankers; (2) whether task-aligned evidence localization better identifies which representations should be preserved under a fixed compression budget; (3) whether instance-specific budgeting improves over uniform and heuristic allocation strategies; and (4) whether these gains persist in long-document and cross-domain settings.

4.1 Experimental Setup

Datasets. We evaluate AREA on four reranking benchmarks. **BRIGHT** serves as our primary public benchmark. Following the BM25 setting of ERank, we retrieve the top-100 candidates using the original query and report nDCG@10 over all 12 domains. We further evaluate cross-domain generalization on five BEIR tasks: Climate-FEVER (CliF.), NFCorpus (NFC.), SciFact (SciF.), TREC-COVID (TREC-C.), and Touché-2020 (TouC.), using top-100 BM25 candidates retrieved with Pyserini.

We additionally use an industrial search benchmark containing 300 queries and 1,728 professionally annotated query–document pairs with four-level relevance judgments. Finally, we construct **BRIGHT-Long** by reconstructing longer candidate documents from BRIGHT and mapping the original relevance annotations to the new corpus, with up to 50 BM25 candidates per query. Detailed dataset construction, statistics, and candidate-retrieval protocols are provided in Appendices D.1, D.2, and D.3.

Baselines, Metrics, and Implementation. We compare AREA with representative LLM rerankers using their public checkpoints and official inference settings. AREA is initialized from Qwen3.5-9B and cold-started with supervised relevance training. Separate models are trained for the public and industrial settings. Full, Cumulative, and Adaptive start from the same pre-RL checkpoint and are optimized under matched RL settings, differing only in the computation-allocation strategy. The localization study instead uses the common pre-RL reranker to isolate the importance predictor. Full training configurations and module specifications are provided in Appendices A.2 and A.3.

We report nDCG@10 on BRIGHT, BRIGHT-Long, and BEIR, and nDCG@10 and nPNR on the industrial benchmark. AREA uses $\ell_c = 8$ and block size $m = 4$; sensitivity to both choices is reported in Appendix B.1. As an importance-only heuristic, k_x retains the minimum set of highest-scoring

Table 1: nDCG@10 (%) and average wall-clock reranking time per query on BRIGHT. Avg. denotes the macro average over 12 domains.

Method	Avg.	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Time/q. (s)↓
BM25	13.70	18.20	27.90	16.50	13.40	10.90	16.30	16.10	24.70	4.30	6.50	7.30	2.10	–
Rank1-7B	18.43	30.51	30.40	16.18	25.05	13.69	16.18	22.64	16.41	15.86	10.71	13.28	10.24	2.924
ReARank-7B	17.57	22.30	26.76	17.24	23.56	16.42	16.12	24.78	28.13	7.16	8.41	10.57	9.38	1.169
ERANK-4B	22.90	30.90	42.12	22.61	27.82	22.12	23.51	25.99	31.73	15.62	11.67	11.11	9.68	1.392
ERANK-14B	23.43	31.05	43.16	23.06	28.93	22.68	25.61	26.29	30.47	17.76	11.65	11.52	8.86	2.945
ERANK-32B	24.58	34.10	43.95	23.84	29.91	22.41	27.54	26.87	33.36	17.43	11.91	11.98	11.65	6.541
ReasonRank-7B	23.86	34.07	43.66	27.01	32.42	27.30	20.95	28.90	26.96	11.89	9.51	12.51	11.12	1.863
ReasonRank-32B	24.72	33.05	43.81	26.39	32.56	29.09	22.64	30.52	30.41	12.75	10.00	12.86	12.59	8.986
AREA-9B (Full)	30.10	39.83	53.83	31.22	34.05	31.68	30.03	33.78	32.91	32.90	14.10	14.94	11.91	0.694
AREA-9B (Cumulative)	29.97	39.95	52.84	31.18	33.70	27.13	31.51	33.17	37.41	32.60	13.30	14.66	12.18	0.562
AREA-9B (Adaptive)	**30.21**	39.49	49.50	31.85	33.68	27.16	30.22	33.10	37.56	40.09	13.31	14.37	12.14	**0.500**

blocks whose cumulative normalized importance reaches k_x; we use $k_x = 0.8$ unless otherwise specified.

System-level efficiency is measured on the same server with $8\times$ NVIDIA A800 80GB GPUs. For BRIGHT and BRIGHT-Long, we report the average wall-clock reranking time per query over the complete candidate set. Detailed timing protocols are provided in Appendix A.4, while controlled model-forward measurements of the final AREA variants are reported in Appendix C.1.

4.2 Main Results on BRIGHT

Table 1 compares AREA with representative LLM rerankers across the 12 BRIGHT domains. The SFT cold-start model obtains 27.53 nDCG@10 before the subsequent ranking optimization. The full-context ranking-optimized checkpoint, AREA (Full), reaches 30.10 nDCG@10, compared with 24.72 for the strongest competing reranker, ReasonRank-32B. The separately optimized Cumulative variant achieves 29.97 nDCG@10, while AREA (Adaptive) reaches 30.21 with an average retention ratio of 0.49. The compressed variants therefore maintain ranking effectiveness comparable to the full-context model under reduced intermediate computation.

At the system level, AREA (Full) requires an average of 0.694 seconds of wall-clock reranking time per query over the complete top-100 candidate set, compared with 0.562 seconds for Cumulative and 0.500 seconds for Adaptive. Adaptive therefore reduces average wall-clock reranking time by 28.0% relative to Full while maintaining comparable ranking effectiveness. The detailed timing protocol is provided in Appendix A.4. Under a matched HuggingFace implementation, the final Full, Cumulative, and Adaptive checkpoints require 89.30, 73.89, and 67.21 ms of model-forward latency per query–document pair, respectively, corresponding to $1.21\times$ and $1.33\times$ speedups for Cumulative and Adaptive over Full (Appendix C.1).

4.3 Task-Aligned Evidence Localization

We isolate evidence localization before ranking-guided RL to evaluate which intermediate representations should be preserved. All methods use the same underlying reranker and compression layer, with random block selection included as an additional control.

Figure 2 compares localization strategies across retention ratios on BRIGHT and Industrial Search. AREA consistently outperforms LTC and random selection across the evaluated range, with larger advantages under stronger compression. As the retention budget increases, AREA approaches full-context effectiveness more rapidly than the alternatives. These results show that task-aligned localization more reliably identifies representations that should be preserved across different compression levels. Controlled forward-pass measurements under a matched HuggingFace implementation further confirm that intermediate compression provides substantial latency reductions; full configuration sensitivity is reported in Appendix B.1.

4.4 Adaptive Budget Allocation

We next evaluate instance-specific budget allocation against fixed, cumulative-importance, and alternative prediction strategies. Except for Shuffled Budget, all variants start from the same pre-RL

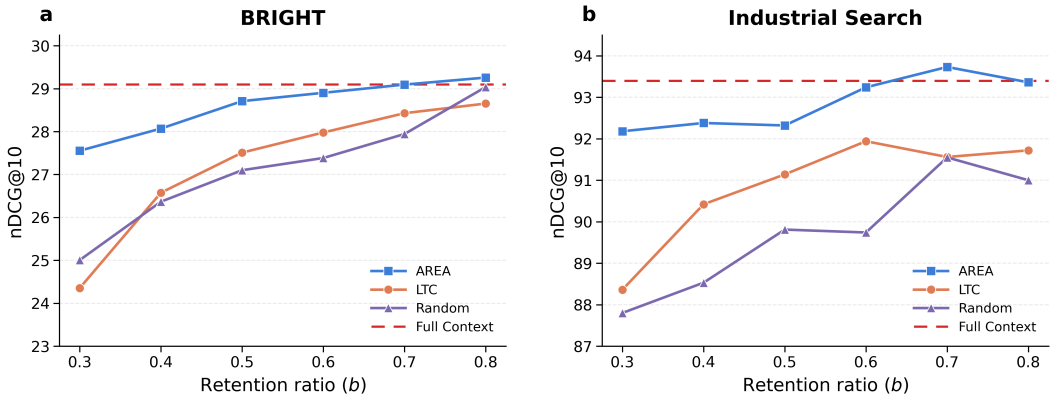


Figure 2: Evidence localization across retention ratios on BRIGHT and Industrial Search before ranking-guided RL. AREA is compared with LTC and random block selection using the same underlying reranker. The dashed line denotes full-context inference.

Table 2: Comparison of budget allocation strategies on BRIGHT and Industrial Search. Ret. denotes average retention. All learned variants start from the same pre-RL checkpoint and use matched RL settings; Shuffled Budget is an inference-time permutation control.

	BRIGHT		Industrial Search		
Method	**Ret.↓**	**nDCG@10↑**	**Ret.↓**	**nDCG@10↑**	**nPNR↑**
Fixed ($b = 0.5$)	0.50	29.21	0.50	92.90	0.8281
Cumulative ($k_x = 0.8$)	0.63	29.97	0.61	94.35	0.8476
Cumulative ($k_x = 0.6$)	0.49	29.33	0.50	93.40	0.8408
Last-token State	0.41	29.04	0.43	92.69	0.8190
Length-only	0.39	28.98	0.40	92.55	0.8144
AREA (Adaptive)	**0.49**	**30.21**	**0.49**	**94.56**	**0.8538**
Shuffled Budget	0.49	29.36	0.49	93.57	0.8329

checkpoint and are optimized under matched RL settings, differing only in the allocation strategy. Shuffled Budget is an inference-time control obtained by permuting AREA's predicted budgets across query–document pairs.

AREA (Adaptive) provides the strongest effectiveness–retention trade-off. At comparable retention, it improves nDCG@10 from 29.33 to 30.21 on BRIGHT and from 93.40 to 94.56 on Industrial Search over Cumulative ($k_x = 0.6$). Compared with Cumulative ($k_x = 0.8$), it achieves comparable or higher effectiveness with substantially lower average retention. Under a matched HuggingFace implementation, Cumulative ($k_x = 0.6$) and Adaptive also exhibit nearly identical model-forward latency (66.98 vs. 67.21 ms), while Adaptive improves nDCG@10 by 0.88 (Appendix C.1). The gain is consistent across three RL-stage random seeds, and paired bootstrap analysis yields positive 95% confidence intervals over both fixed-ratio and retention-matched cumulative allocation (Appendix C.4).

Under the same budget-learning setup, using either the last-token hidden state or document length as the predictor input yields lower effectiveness and converges to nearly constant budgets across instances (Appendix C.2). AREA instead conditions the budget predictor on the distribution of localization scores. Additional analyses validate the candidate-wise sufficiency approximation under joint compression and evaluate the fidelity of directional budget search against exhaustive budget evaluation (Appendices C.3 and C.5).

4.5 Long-Document Reranking

We further evaluate AREA on BRIGHT-Long, where longer candidate documents increase the cost of propagating full-resolution representations through the reranker. Table 3 reports ranking effectiveness and average wall-clock reranking time per query for representative LLM rerankers.

Table 3: nDCG@10 (%) and average wall-clock reranking time per query on BRIGHT-Long. Avg. denotes the macro average over 12 domains.

Method	Avg.	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Time/q. (s)↓
Rank1-7B	26.23	48.47	42.21	18.42	23.56	25.26	16.65	30.29	23.80	43.43	11.62	16.24	14.77	9.065
ReARank-7B	26.07	43.01	42.07	21.02	22.11	26.12	19.46	39.04	31.75	32.53	9.86	12.85	12.98	1.288
ERANK-4B	34.01	50.87	49.91	25.36	31.32	36.18	29.73	43.51	**39.43**	54.70	15.56	16.73	14.83	1.073
ERANK-14B	34.76	50.14	49.86	27.88	33.39	37.55	29.34	43.63	39.08	59.45	16.48	16.47	13.82	2.598
ReasonRank-32B	31.55	57.24	45.15	23.90	27.68	34.90	26.48	38.80	32.53	45.27	12.67	**17.50**	**16.52**	6.500
AREA-9B (Full)	**38.06**	57.03	53.07	29.05	**34.53**	39.20	**32.78**	47.67	35.75	**77.33**	**16.95**	17.42	15.96	0.827
AREA-9B (Cumulative)	37.36	**57.64**	53.04	30.20	33.23	38.97	32.40	**48.10**	33.77	72.91	15.29	17.42	15.31	0.704
AREA-9B (Adaptive)	37.64	56.95	**53.26**	**30.48**	33.51	**40.14**	32.43	46.83	35.16	73.70	16.05	17.42	15.72	**0.581**

Table 4: Cross-domain results on five BEIR retrieval tasks. We report nDCG@10 (%) and the macro average.

Method	Avg.	CliF.	NFC.	SciF.	TREC-C.	TouC.
BM25	44.06	16.50	32.20	67.90	59.50	44.20
Rank1-7B	46.80	16.70	35.60	76.80	79.80	25.10
ReARank-7B	49.04	21.90	32.70	74.80	79.70	36.10
ERANK-4B	47.10	17.80	35.10	73.10	79.60	29.90
ERANK-14B	48.16	18.50	35.00	74.20	81.80	31.30
ERANK-32B	48.76	18.70	36.70	73.90	82.40	32.10
ReasonRank-7B	50.54	22.00	39.40	75.20	82.10	34.00
ReasonRank-32B	50.32	20.60	**40.10**	77.10	83.40	30.40
AREA-9B (Full)	51.32	23.14	37.65	77.87	**83.98**	33.94
AREA-9B (Cumulative)	51.44	22.21	37.26	79.22	82.97	**35.52**
AREA-9B (Adaptive)	**51.87**	**23.79**	37.52	**79.66**	83.07	35.32

On BRIGHT-Long, AREA maintains strong ranking effectiveness while reducing wall-clock reranking time. AREA (Full) achieves an average nDCG@10 of 38.06 with 0.827 seconds of reranking time per query. AREA (Adaptive) requires 0.581 seconds per query, 29.7% less than AREA (Full), while retaining 37.64 nDCG@10. AREA (Cumulative) provides an intermediate operating point at 37.36 nDCG@10 and 0.704 seconds per query.

4.6 Cross-Domain Generalization

Table 4 reports cross-domain results on five BEIR tasks. AREA (Full), AREA (Cumulative), and AREA (Adaptive) achieve macro-average nDCG@10 scores of 51.32, 51.44, and 51.87, respectively. The compressed variants therefore maintain effectiveness comparable to the separately optimized full-context model, with AREA (Adaptive) remaining competitive across individual datasets. Together with the BRIGHT and BRIGHT-Long results, these findings support the generalization of adaptive relevance evidence allocation across retrieval domains.

5 Conclusion

We introduced AREA, an adaptive relevance evidence allocation framework for efficient LLM reranking. Rather than applying a uniform compression policy to all query–document pairs, AREA jointly learns where relevance-bearing representations should be preserved and how much evidence each instance requires for deeper computation. Both decisions are supervised by signals derived from relevance learning itself, enabling task-aligned evidence localization and instance-specific budgeting without explicit token- or budget-level annotations. Experiments across BRIGHT, industrial search, long-document reranking, and cross-domain evaluation show that task-aligned localization preserves ranking effectiveness more reliably under constrained budgets, while adaptive budgeting further improves the effectiveness–efficiency trade-off. These results suggest that efficient reranking should allocate computation according to the distribution and sufficiency of relevance evidence, rather than processing every input under a uniform retention budget.

AI USE STATEMENT

Generative AI tools were used to assist with brainstorming and refining research methodology and experimental design, literature discovery and summarization, software development and debugging, and drafting and editing portions of the manuscript. All AI-assisted suggestions were critically reviewed by the authors. Research decisions, method implementations, experimental results, and interpretations were independently verified by the authors, and relevant literature suggested by AI tools was checked against the original sources. Generative AI tools were not used as a substitute for conducting experiments or validating empirical results. The authors take full responsibility for the final content of this work, including all text, claims, code, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies efficient relevance reranking and does not involve interventions on human subjects or high-stakes decision making. Experiments on public benchmarks use existing datasets under their established research settings. The internal industrial benchmark consists of proprietary search data with human relevance annotations and is used only for research evaluation; we report only aggregate experimental results and do not release the underlying proprietary data. As with other learning-based ranking systems, the proposed method may inherit biases present in the training data or pretrained language model, and its effectiveness may vary across domains, languages, and retrieval settings. We therefore view the method as a general relevance modeling and efficiency technique rather than a system intended for sensitive or consequential applications without additional domain-specific evaluation. All experiments and analyses were conducted in accordance with applicable data-access and research-integrity requirements.

REPRODUCIBILITY STATEMENT

We provide the methodological and experimental details necessary to reproduce the main results throughout the paper and appendix. The main text specifies the model architecture, relevance evidence localization and compression mechanisms, adaptive budget learning procedure, and training objectives. The appendix further documents implementation details, training and inference hyperparameters, dataset construction and preprocessing, evaluation protocols, and additional ablation studies. Experiments on public benchmarks use publicly available datasets and established evaluation metrics. For the proprietary industrial benchmark, the underlying data cannot be released due to data-access restrictions; however, we provide its construction procedure, annotation protocol, and evaluation setting to clarify the experimental methodology. Additional implementation details and reproducibility materials are included in the supplementary material where permitted by data and licensing constraints.

REFERENCES

Yuzheng Cai, Yanzhao Zhang, Dingkun Long, Mingxin Li, Pengjun Xie, and Weiguo Zheng. Erank: Fusing supervised fine-tuning and reinforcement learning for effective and efficient text reranking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 30121–30129, 2026.

Nayoung Choi, Youngjune Lee, Gyu-Hwung Cho, Haeyu Jeong, Jungmin Kong, Saehun Kim, Keunchan Park, Sarah Cho, Inchang Jeong, Gyohee Nam, Sunghoon Han, Wonil Yang, and Jaeho Choi. RRADistill: Distilling LLMs' passage ranking ability for long-tail queries document re-ranking on a search engine. In Franck Dernoncourt, Daniel Preoţiuc-Pietro, and Anastasia Shimorina (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 627–641, Miami, Florida, US, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-industry.46. URL https://aclanthology.org/2024.emnlp-industry.46/.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. PoWER-BERT: Accelerating BERT inference via progressive word-

vector elimination. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3690–3699. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/goyal20a.html.

Yue Guan, Zhengyi Li, Jingwen Leng, Zhouhan Lin, and Minyi Guo. Transkimmer: Transformer learns to layer-wise skim. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7275–7286, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.502. URL https://aclanthology.org/2022.acl-long.502/.

Gyuwan Kim and Kyunghyun Cho. Length-adaptive transformer: Train once with length drop, use anytime with search. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6501–6511, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.508. URL https://aclanthology.org/2021.acl-long.508/.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. ReasonRank: Empowering passage ranking with strong reasoning ability. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 22007–22031, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1005. URL https://aclanthology.org/2026.acl-long.1005/.

Xuan Lu, Haohang Huang, Yingqi Fan, Junlong Tong, Yuxuan Zhang, Ping Nie, Rui Meng, and Xiaoyu Shen. Comprank: Efficient llm reranking via token-level compression and decoding-free scoring. *arXiv preprint arXiv:2606.11700*, 2026.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 2421–2425, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657951. URL https://doi.org/10.1145/3626772.3657951.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pre-trained sequence-to-sequence model. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 708–718, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.63. URL https://aclanthology.org/2020.findings-emnlp.63/.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 1504–1518, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.97. URL https://aclanthology.org/2024.findings-naacl.97/.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Liu Haisu, Quan Shi, Zachary Siegel, Michael Tang, et al. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. In *International Conference on Learning Representations*, volume 2025, pp. 48941–48991, 2025.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is ChatGPT good at search? investigating large language models as re-ranking agents. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 14918–14937, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.923. URL https://aclanthology.org/2023.emnlp-main.923/.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Advances in Neural Information Processing Systems (NeurIPS), Track on Datasets and Benchmarks*, volume 1, 2021.

Yu Wang, Shengyao Zhuang, Xueguang Ma, Zongyu Wu, Jimmy Lin, Vivek Srikumar, and Zhichao Xu. Tevatron-elastic: A unified abstraction for training elastic retrievers and rerankers. *arXiv preprint arXiv:2608.08809*, 2026.

Daowei Xie, Jing Luo, and Xinhui Tu. Serank: Sentence-level evidence selection for llm-based document reasoning reranking. *Expert Systems with Applications*, 333:134288, 2027. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2026.134288. URL https://www.sciencedirect.com/science/article/pii/S0957417426031945.

Deming Ye, Yankai Lin, Yufei Huang, and Maosong Sun. TR-BERT: Dynamic token reduction for accelerating BERT inference. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5798–5809, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.463. URL https://aclanthology.org/2021.naacl-main.463/.

Le Zhang, Bo Wang, Xipeng Qiu, Siva Reddy, and Aishwarya Agrawal. REARANK: Reasoning re-ranking agent via reinforcement learning. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 2458–2471, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.125. URL https://aclanthology.org/2025.emnlp-main.125/.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Mike Bendersky. Rankt5: Fine-tuning t5 for text ranking with ranking losses. In *Proc. of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, 2023.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 358–370, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-short.31. URL https://aclanthology.org/2024.naacl-short.31/.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, pp. 38–47, New York, NY, USA, 2024b. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657813. URL https://doi.org/10.1145/3626772.3657813.

Shengyao Zhuang, Xueguang Ma, Zheng Yao, Shuai Wang, Bevan Koopman, Jimmy Lin, and Guido Zuccon. Rank-r1: Enhancing reasoning in llm-based document rerankers via reinforcement learning. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, pp. 4419–4425, New York, NY, USA, 2026a. Association for Computing Machinery. ISBN 9798400725999. doi: 10.1145/3805712.3809961. URL https://doi.org/10.1145/3805712.3809961.

Shengyao Zhuang, Zhichao Xu, and Ivano Lauriola. Layer-wise token compression for efficient document reranking. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '26, pp. 4426–4432, New York, NY, USA, 2026b. Association for Computing Machinery. ISBN 9798400725999. doi: 10.1145/3805712.3809871. URL https://doi.org/10.1145/3805712.3809871.

A Experimental Details

A.1 Cold-Start Supervised Fine-Tuning

We train separate AREA models for the public English and industrial Chinese settings. The AREA student is initialized from Qwen3.5-9B and first undergoes supervised fine-tuning (SFT) for relevance judgment before task-aligned localization and ranking-guided adaptive budgeting. For relevance distillation, we additionally initialize a teacher from Qwen3.5-27B and apply the same cold-start SFT procedure. The resulting 27B model is kept fixed during distillation.

For the public model, the SFT corpus contains 8,866 queries and 32,579 query–document pairs, covering approximately 68% of the ReasonRank training set. For the industrial model, we use 25,102 query–document pairs sampled from historical search data. The two models are trained independently.

For each query–document pair, GPT-5.4 generates a four-level relevance score $y \in \{0, 1, 2, 3\}$ together with a structured chain-of-thought rationale. The rationale follows a fixed two-stage format. It first identifies the query intent, expected answer type, core object and scope, key answer elements, and relevant constraints. It then evaluates the candidate document with respect to intent alignment, entity and key-information matching, completeness, and the proportion of relevant content before producing the final relevance score. The four labels correspond to fully satisfied, mostly satisfied, partially satisfied, and not satisfied, respectively.

Given a query q, candidate document d, generated rationale c, and relevance label y, the cold-start model is trained with the standard autoregressive objective

$$\mathcal{L}_{\mathrm{SFT}} = -\mathbb{E}_{(q,d,c,y)} \log p_\theta(c, y \mid q, d).$$

At inference time, the model receives only the query and candidate document. No evaluation examples from BRIGHT, BEIR, BRIGHT-Long, or the industrial test benchmark are included in the SFT data.

A.2 Training Configuration

AREA is trained in four stages: cold-start supervised fine-tuning, relevance distillation, task-aligned evidence localization, and ranking-guided adaptive budgeting. AREA uses Qwen3.5-9B as the trainable backbone throughout all stages. Relevance distillation additionally uses the fixed SFT-trained Qwen3.5-27B teacher described in Appendix A.1. Table 5 summarizes the main optimization settings of the trainable AREA model.

The cold-start stage performs full-parameter supervised fine-tuning with a global batch size of 128 for five epochs. We use cosine learning-rate decay from 10^{-5} to 10^{-6} with a 5% warmup, a maximum sequence length of 7680 tokens, and BF16 training. No gradient clipping is applied.

The relevance-distillation stage uses 64 prompts per trainer step and samples four trajectories per prompt. The actor is optimized for one epoch with a constant learning rate of 10^{-6}. The subsequent localization stage freezes the reranker and updates only the importance predictor. The predictor is trained for two epochs with an effective batch size of 64; samples longer than 3072 tokens are filtered rather than truncated.

During ranking-guided adaptive budgeting, each training batch contains 32 query–document instances with four rollouts per instance. The trainable reranker layers use a learning rate of 10^{-6}, while the budget predictor is optimized separately at 10^{-4}. The localization module remains fixed throughout this stage.

Adaptive Budgeting Hyperparameters. Table 6 lists the hyperparameters specific to counterfactual budget learning.

Counterfactual budget probes are performed on the feasible block lattice rather than by applying a fixed continuous step. The search starts with an integer block increment corresponding approximately to a 0.05 change in retention and expands geometrically when the current direction remains unbracketed, up to approximately 0.20 retention.

Table 5: Training configuration of AREA. $N \times G$ denotes N prompts/instances with G rollouts. For stages with multiple optimizers, learning rates are reported separately.

Setting	Cold-start SFT	Distillation	Localization	Budget / RL
Optimizer	AdamW	AdamW	AdamW	AdamW
Learning rate	1×10^{-5}	1×10^{-6}	1×10^{-3}	1×10^{-6} / 1×10^{-4}
Epochs	5	1	2	3
Batch size	128	64×4	64	32×4
Max. sequence length	7680	3584	3072	2560
LR schedule	Cosine	Constant	Cosine	Constant
Warmup ratio	0.05	0	0.05	0
Weight decay	0.01	0.01	0	0.01
Precision	BF16	BF16	BF16 / FP32	BF16 / FP32
Gradient clipping	None	1.0	1.0	1.0
Random seed	42	42	1234	42

Table 6: Hyperparameters for ranking-guided adaptive budget learning.

Hyperparameter	Value
Rollouts per instance G	4
KL coefficient λ_{KL}	0.001
Sufficiency tolerance τ	0.001
Minimum budget $b_{\min}$	0.25
Maximum budget $b_{\max}$	1.00
Initial search increment	≈ 0.05 retention
Maximum search increment	≈ 0.20 retention
Budget-head learning rate	1×10^{-4}
Budget-head loss	MSE + upper-bound hinge

The budget predictor receives two forms of counterfactual supervision. A confirmed sufficient configuration provides an MSE regression target, whereas an observed valid upper bound provides a one-sided squared-hinge constraint:

$$\mathcal{L}_{\mathrm{budget}} = \frac{\sum_i w_i \left[\mathbf{1}_{t_i} (\hat{b}_i - b_i^{\mathrm{target}})^2 + \mathbf{1}_{u_i} [\hat{b}_i - b_i^{\mathrm{upper}}]_+^2 \right]}{\sum_i w_i}.$$

Here, b_i^{target} denotes the confirmed sufficient budget, b_i^{upper} denotes a valid upper bound, and $\mathbf{1}_{t_i}$ and $\mathbf{1}_{u_i}$ indicate their availability, respectively. We set $w_i = \exp(-5\,\Delta_i)\,0.9^{s_i}$, where Δ_i is the search-bracket width and s_i is the label staleness.

The lower-censored term supported by the implementation is disabled in all reported experiments. If neither a finite target nor an upper bound is available, no budget-head optimizer step is performed.

During the final RL stage, the localization predictor is frozen. We also freeze the token embeddings, Transformer layers below the compression layer $[0, \ell_c)$, and the language-model head. Layers at and above $\ell_c = 8$ remain trainable. The budget predictor uses a separate optimizer and does not receive gradients from the actor objective.

A.3 Module Architecture and Compression Configuration

Importance Predictor. AREA uses a lightweight token-wise predictor at the compression layer $\ell_c = 8$. Given an intermediate representation $h_i \in \mathbb{R}^d$, the predictor computes

$$s_i = W_2\,\mathrm{GELU}\,(W_1\,\mathrm{LayerNorm}(h_i) + b_1) + b_2,$$

where $W_1 : \mathbb{R}^d \to \mathbb{R}^{256}$ and $W_2 : \mathbb{R}^{256} \to \mathbb{R}$. We use standard LayerNorm and no dropout. For a block B_j, the block-level score is the mean of its token scores,

$$s_j = \frac{1}{|B_j|} \sum_{i \in B_j} s_i.$$

The same predictor and block aggregation rule are used during localization training and compressed inference.

All reported experiments use the `whole_minus_tail` configuration. Specifically, for a prompt of length L, the compressible span is $[0, L-32)$, while the final 32 prompt tokens are always preserved at full resolution. Thus, instruction, query, and document representations within the compressible prefix are jointly scored and may be pooled; no additional token-type-specific protection rule is applied.

Budget Predictor. The budget predictor operates on the distribution of block importance scores. For an instance with M valid blocks, we first normalize the block scores along the sequence dimension,

$$x_j = \frac{\exp(s_j)}{\sum_{r=1}^{M} \exp(s_r)}, \qquad j = 1, \ldots, M.$$

The normalized scores remain in their original block order and are right-padded with zeros to a fixed dimension of 632, corresponding to the maximum compressible span under a prompt length of 2560, a 32-token protected tail, and block size $m = 4$.

The resulting vector $\tilde{x} \in \mathbb{R}^{632}$ is processed by a three-layer MLP,

$$632 \to 1024 \to 256 \to 1,$$

with GELU activations and no normalization or dropout layers. Its scalar output is mapped to the admissible retention interval as

$$\hat{b} = b_{\min} + (b_{\max} - b_{\min})\sigma\left(f_\psi(\tilde{x})\right),$$

where $b_{\min} = 0.25$ and $b_{\max} = 1.0$ in the adaptive setting.

The first two linear layers are initialized from $\mathcal{N}(0, 0.02^2)$, while the final projection is initialized to zero. Consequently, the initial predictor outputs the midpoint of the admissible interval, $\hat{b} = 0.625$, for all instances. The budget predictor contains approximately 0.91M parameters and is evaluated in FP32.

Protect-or-Pool Execution. The compression layer partitions the compressible span into non-overlapping blocks of size $m = 4$. A protected block retains all of its token representations, whereas an unprotected block is replaced by the arithmetic mean of its representations. After this decision, all retained and pooled representations are restored to their original sequence order; block importance affects only which blocks are protected and does not reorder the sequence.

The implementation also supports a final partial block. Let n_{span} be the number of compressible tokens, $M = \lceil n_{\text{span}}/m \rceil$, and L_j the actual number of tokens in block B_j. Let $\pi_1, \ldots, \pi_M$ denote the blocks sorted by decreasing importance. Given a continuous budget $\hat{b}$, we first obtain a token target

$$T = \text{round}(\hat{b}\, n_{\text{span}}),$$

and choose the largest number of protected blocks satisfying

$$P = \max\left\{ k : M + \sum_{r=1}^{k} (L_{\pi_r} - 1) \le T \right\}.$$

The resulting executed retention ratio is therefore

$$b_{\text{exec}} = \frac{M + \sum_{r=1}^{P} (L_{\pi_r} - 1)}{n_{\text{span}}}.$$

When all blocks contain exactly m tokens, this reduces to the expression given in Section 3.2,

$$b_{\text{exec}} = \frac{M + P(m-1)}{mM}.$$

Thus, continuous budget predictions are mapped to the largest feasible Protect-or-Pool configuration that does not exceed the corresponding token budget.

Table 7: Architecture and compression configuration used in all AREA experiments.

Configuration	Value
Compression layer ℓ_c	8
Block size m	4
Compressible span	Prompt except final 32 tokens
Importance-head hidden size	256
Importance-head normalization	LayerNorm
Importance-head activation	GELU
Budget-head input dimension	632
Budget-head hidden dimensions	1024, 256
Budget-head activation	GELU
Budget range	$[0.25, 1.0]$
Protected tail	32 tokens

Sequence Reconstruction. After compression, each shortened sequence is assigned contiguous positions $0, \ldots, L' - 1$ and right-padded to the maximum compressed length within the batch. We rebuild the attention mask, cache positions, position identifiers, and rotary embeddings for the shortened sequence. Because Qwen3.5 interleaves full-attention and linear-attention layers, we reconstruct both the causal attention mask and the linear-attention mask, and each subsequent layer uses the mask corresponding to its layer type.

All reported experiments use cache-free full-sequence forward passes. Accordingly, upper linear-attention layers operate directly on the compressed sequence; no existing recurrent state, convolution state, or KV cache is pooled or remapped. This implementation therefore concerns intermediate compression during full-sequence reranking rather than compression of an already populated generation cache.

A.4 Inference and Efficiency Measurement

Inference Setup. Controlled model-forward latency measurements in Appendices B.1 and C.1 use a single NVIDIA A800 80GB GPU with batch size one and matched input settings. We use PyTorch 2.10, CUDA 12.8, and vLLM 0.18.1. Baseline rerankers are evaluated using their official checkpoints, prompts, inference scripts, and scoring rules. We set temperature to 0 for all models and otherwise follow the model-specific inference configurations, including context truncation and maximum response length, used by the corresponding methods.

AREA predicts relevance over the four ordinal labels $\{0, 1, 2, 3\}$. At inference time, the ranking score is computed as the expectation of the relevance distribution,

$$s(q, d) = \sum_{r=0}^{3} r \, p(r \mid q, d).$$

Although the supervised training targets contain both a relevance label and a reasoning rationale, inference uses only the relevance label. The implementation directly computes the four relevance-digit logits and therefore does not require autoregressive chain-of-thought generation.

For compressed AREA variants, the importance and budget predictors are executed within the same model forward pass. After obtaining the hidden states at layer $\ell_c = 8$, the importance predictor first computes token and block scores, after which the budget predictor consumes the normalized block-score distribution before Protect-or-Pool compression is applied. Neither predictor requires an additional model forward pass.

AREA Checkpoints. AREA (Full), AREA (Cumulative), and AREA (Adaptive) are initialized from the same pre-RL checkpoint and optimized separately under their respective computation-allocation strategies using matched RL data, rewards, and optimization settings. AREA (Full) disables intermediate compression, AREA (Cumulative) applies cumulative-importance allocation, and AREA (Adaptive) uses the learned instance-specific budget predictor. All three variants use the same input format, relevance scoring rule, and inference output format.

BRIGHT Efficiency Measurement. For BRIGHT, each query reranks the complete top-100 BM25 candidate set. We measure the total wall-clock inference time over all 1,384 queries and report the average reranking time per query,

$$T_{\text{query}} = \frac{T_{\text{inference}}}{1384}.$$

Timing starts after model loading and ends after the final candidate is processed. Tokenization, retrieval, and metric computation are excluded. All methods use eight physical GPUs.

For the controlled experiment in Appendix B.1, Full Context, LTC, and AREA are evaluated using the same HuggingFace implementation on a single NVIDIA A800 80GB GPU with batch size one and matched input settings. We report pure model-forward time per query–document pair, excluding tokenization, retrieval, and metric computation. For compressed AREA variants, controlled latency includes importance prediction, budget prediction where applicable, and Protect-or-Pool execution.

BRIGHT-Long Efficiency Measurement. BRIGHT-Long contains 1,193 queries, each reranking up to 50 BM25 candidates. We use the same timing protocol as for BRIGHT and report the average wall-clock reranking time per query,

$$T_{\text{query}} = \frac{T_{\text{inference}}}{1193}.$$

Timing starts after model loading and ends after the final candidate is processed. Tokenization, retrieval, and metric computation are excluded.

All BRIGHT-Long experiments use $8\times$ NVIDIA A800 80GB GPUs. The uncompressed scorer uses vLLM where supported. Because native vLLM does not expose the intermediate hidden-state rewriting required by AREA compression, the Cumulative and Adaptive variants use custom PyTorch forward passes in the same software and hardware environment. Their importance prediction, budget prediction, and Protect-or-Pool operations are included in the reported inference time.

For all BRIGHT-Long results reported in Table 3, AREA (Full), AREA (Cumulative), and AREA (Adaptive) use the same maximum formatted prompt length of 2,560 tokens. Documents are truncated when necessary to satisfy this limit. Such truncation is rare, affecting fewer than 0.5% of query–document pairs. All three variants therefore operate under the same input-length constraint. The reported wall-clock times characterize their respective complete inference configurations. Controlled same-implementation measurements of the final AREA variants are reported in Appendix C.1.

B Additional Analysis of Evidence Localization

B.1 Compression Configuration and Controlled Forward Latency

We examine the sensitivity of AREA to the compression layer ℓ_c and block size m before ranking-guided optimization. All compressed configurations use the same pre-RL reranker and a fixed retention ratio of $b = 0.5$, allowing the effect of the compression configuration to be isolated. We evaluate $\ell_c \in \{4, 8, 16\}$ and $m \in \{4, 8, 16\}$ on both BRIGHT and Industrial Search. LTC is reproduced on the same underlying reranker at $\ell_c = 8$.

To isolate the computational effect of intermediate compression from differences in serving engines, all latency measurements in this experiment use the same HuggingFace implementation on a single NVIDIA A800 80GB GPU with batch size one. We report pure model-forward time per query–document pair, excluding tokenization, retrieval, and metric computation. AREA latency includes the importance-predictor computation.

Table 8 shows a consistent effectiveness–latency trade-off across compression layers. Earlier compression at $\ell_c = 4$ provides the lowest latency but incurs a larger loss in ranking effectiveness. Moving compression to $\ell_c = 16$ recovers more of the full-context effectiveness, but substantially reduces the computational benefit. We therefore use $\ell_c = 8$ as the default intermediate operating point. At this layer, $m = 4$ provides the strongest ranking effectiveness among the tested block sizes on both benchmarks, with similar latency to larger blocks.

Table 8: Sensitivity to compression layer ℓ_c and block size m before ranking-guided optimization. All compressed methods use a fixed retention ratio of 0.5. Latency is pure model-forward time per query–document pair, measured with the same HuggingFace implementation on a single NVIDIA A800 80GB GPU with batch size one. nDCG@10 is reported in percentage points.

				BRIGHT		Industrial Search		
Method	ℓ_c	m	**Ret.**	**nDCG@10↑**	**Lat. (ms)↓**	**nDCG@10↑**	**nPNR↑**	**Lat. (ms)↓**
Full Context	–	–	1.00	29.10	90.59	93.38	0.8281	69.85
LTC	8	–	0.50	27.72	70.12	90.87	0.8101	57.05
AREA	4	4	0.50	27.76	65.66	91.04	0.8146	55.40
AREA	4	8	0.50	28.26	64.79	91.06	0.8182	55.88
AREA	4	16	0.50	27.08	64.77	90.17	0.8139	55.27
AREA†	8	4	0.50	28.81	68.94	92.32	0.8261	57.76
AREA	8	8	0.50	28.45	68.87	91.82	0.8279	57.47
AREA	8	16	0.50	28.55	70.36	91.91	0.8239	57.34
AREA	16	4	0.50	29.29	77.92	92.44	0.8239	63.84
AREA	16	8	0.50	29.24	78.11	92.52	0.8246	63.61
AREA	16	16	0.50	29.29	77.92	92.15	0.8235	63.79

† Default configuration used in the main experiments.

Under the default configuration ($\ell_c = 8, m = 4$), AREA achieves 28.81 nDCG@10 on BRIGHT and 92.32 on Industrial Search, compared with 27.72 and 90.87 for LTC at the same retention ratio. Its forward latency is 68.94 ms on BRIGHT and 57.76 ms on Industrial Search, corresponding to $1.31\times$ and $1.21\times$ speedups over full-context inference, respectively. Thus, the localization improvement is obtained while retaining the computational benefit of intermediate compression.

The full-context row uses the same pre-RL reranker and is included only as a reference operating point; it therefore differs from the final ranking-optimized full-context model reported in the main results.

B.2 Magnitude versus Signed Compression Sensitivity

Section 3.3 defines block importance using the magnitude of a first-order local surrogate for the counterfactual effect of compression. Here we further motivate the use of magnitude and compare it with signed supervision.

Signed First-Order Surrogate Effect. For a block B_j, consider the same-length mean-replacement surrogate used in Section 3.3, in which each hidden representation $h_k^{(\ell_c)}$ in B_j is replaced by the block mean $\bar{h}_j$ while keeping the sequence length and positions fixed. Let

$$\delta h_k = \bar{h}_j - h_k^{(\ell_c)}.$$

With

$$g_k = \frac{\partial \mathcal{L}_{\text{dist}}}{\partial h_k^{(\ell_c)}},$$

the signed first-order score of this local perturbation is

$$\widetilde{\Delta}_j = \sum_{k \in B_j} g_k^\top \delta h_k.$$

To make the dependence of its sign explicit, consider the downstream relevance logits z and let

$$J_k = \frac{\partial z}{\partial h_k^{(\ell_c)}}, \qquad r = \frac{\partial \mathcal{L}_{\text{dist}}}{\partial z}.$$

By the chain rule,

$$g_k = J_k^\top r.$$

Therefore,

$$\widetilde{\Delta}_j = \sum_{k \in B_j} \delta h_k^\top J_k^\top r = \left\langle \sum_{k \in B_j} J_k \delta h_k,\, r \right\rangle.$$

Defining

$$v_j = \sum_{k \in B_j} J_k \delta h_k,$$

we obtain

$$\widetilde{\Delta}_j = \langle v_j, r \rangle.$$

This decomposition separates two factors. The term v_j characterizes how the local mean-replacement perturbation affects the downstream prediction, whereas r specifies the direction in which that perturbation affects the distillation objective. The residual r depends on the student–teacher predictive discrepancy and is therefore determined by training-time supervision that is not provided as input to the inference-time importance predictor.

Consequently, the sign of $\widetilde{\Delta}_j$ is not, in general, determined by $h^{(\ell_c)}$ alone. For example, holding the intermediate representations and downstream perturbation v_j fixed, reversing the supervision residual from r to $-r$ changes

$$\widetilde{\Delta}_j = \langle v_j, r \rangle \quad \text{to} \quad -\widetilde{\Delta}_j,$$

while the input available to the importance predictor remains unchanged. Thus, a predictor based only on inference-time intermediate representations cannot in general recover the direction of this supervision-dependent first-order effect.

Magnitude as Compression Sensitivity. Our objective is therefore not to predict whether pooling a block locally increases or decreases the training loss, but to identify blocks whose compression substantially perturbs the relevance objective. We define

$$I_j = |\widetilde{\Delta}_j| = \left| \sum_{k \in B_j} \left(h_k^{(\ell_c)} - \bar{h}_j \right)^\top g_k \right|.$$

Taking the magnitude removes the sign ambiguity above: under the transformation $r \to -r$, the signed effect reverses whereas $|\langle v_j, r \rangle|$ remains unchanged. The resulting target should therefore be interpreted as *compression sensitivity*, rather than as the direction of beneficial or harmful relevance evidence.

We emphasize that the magnitude is still a task-derived quantity and need not be exactly recoverable from the intermediate representation alone. Instead, discarding the supervision-dependent direction produces a more stable target for learning which blocks are sensitive to compression from inference-time representations.

Empirical Comparison. We compare magnitude-based supervision with a signed counterpart under the same fixed retention ratio of 0.5. For both variants, the block-level first-order effects are converted into supervision distributions using a softmax over blocks. The magnitude variant uses

$$\pi_j^{\text{mag}} = \frac{\exp(|\Delta_j|)}{\sum_r \exp(|\Delta_r|)},$$

whereas the signed variant retains the direction of the first-order effect,

$$\pi_j^{\text{signed}} = \frac{\exp(\Delta_j)}{\sum_r \exp(\Delta_r)}.$$

The two variants otherwise use the same soft-label cross-entropy objective, model initialization, optimization settings, and compression configuration.

As shown in Table 9, signed supervision reduces nDCG@10 from 28.81 to 27.67 on BRIGHT. The same trend is observed on Industrial Search, where nDCG@10 decreases from 92.32 to 91.06 and nPNR from 0.8261 to 0.7963. These results are consistent with the analysis above: using the magnitude of compression sensitivity provides a more reliable localization target than retaining its supervision-dependent direction.

Table 9: Comparison of magnitude- and signed-sensitivity supervision under a fixed retention ratio of 0.5. Both variants convert their respective first-order scores into block-level target distributions using softmax; all other localization and inference settings are identical.

Localization Target	BRIGHT	Industrial Search	
	nDCG@10	nDCG@10	nPNR
Signed	27.67	91.06	0.7963
Magnitude (AREA)	**28.81**	**92.32**	**0.8261**

B.3 Fidelity of the First-Order Approximation

Exact Counterfactual Evaluation. For the exact importance I_j^*, we apply the same Protect-or-Pool operation used during compressed inference. Specifically, block B_j is replaced by a single mean-pooled representation, thereby shortening the sequence, after which positions, rotary embeddings, and attention masks are reconstructed as described in Appendix A.3. We then run the remaining layers on this shortened sequence and recompute the distillation objective. Thus, I_j^* measures the effect of the actual dimension-changing compression operator, rather than a same-length mean-replacement perturbation.

Section 3.3 uses a first-order approximation to estimate the counterfactual effect of pooling each block. Exact evaluation requires an additional compressed forward pass for every candidate block, whereas the approximation computes all block-level targets from a single forward–backward pass.

For block B_j, the exact counterfactual importance is

$$I_j^* = \left| \mathcal{L}_{\text{dist}}^{\text{pool}(B_j)} - \mathcal{L}_{\text{dist}}^{\text{full}} \right|,$$

while AREA uses the first-order score derived from the same-length mean-replacement surrogate,

$$I_j = \left| \sum_{k \in B_j} \left(h_k^{(\ell_c)} - \bar{h}_j \right)^\top g_k \right|, \qquad g_k = \frac{\partial \mathcal{L}_{\text{dist}}}{\partial h_k^{(\ell_c)}}.$$

Zero-Sum Structure of Pooling Perturbations. The first-order score has a useful structure induced by mean pooling. Let

$$S_j = \sum_{k \in B_j} \left(h_k^{(\ell_c)} - \bar{h}_j \right)^\top g_k, \qquad I_j = |S_j|.$$

Because mean pooling preserves the block mean,

$$\sum_{k \in B_j} \left(h_k^{(\ell_c)} - \bar{h}_j \right) = 0.$$

Hence, letting $\bar{g}_j = |B_j|^{-1} \sum_{k \in B_j} g_k$,

$$S_j = \sum_{k \in B_j} \left(h_k^{(\ell_c)} - \bar{h}_j \right)^\top (g_k - \bar{g}_j).$$

Thus, any gradient component shared uniformly across all tokens in the block cancels exactly. Equivalently,

$$S_j = \frac{1}{2|B_j|} \sum_{k,r \in B_j} \left(h_k^{(\ell_c)} - h_r^{(\ell_c)} \right)^\top (g_k - g_r).$$

The approximation therefore depends on the alignment between within-block variation in representations and within-block variation in task sensitivity, rather than on their common components. In particular, if all tokens in a block have identical local gradients, mean pooling has zero first-order effect.

We evaluate the fidelity of the first-order approximation on 100 randomly sampled query–document pairs from each benchmark. For each pair, we first measure score-level agreement using the Pearson

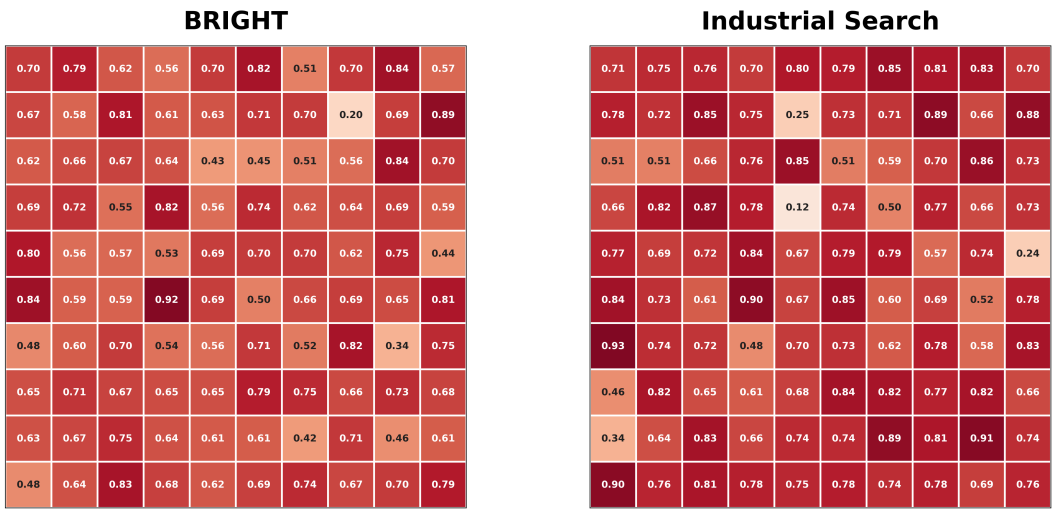


Figure 3: Per-instance score-level fidelity of the first-order surrogate to exact counterfactual compression importance. Exact importance is computed using the actual Protect-or-Pool operator, including sequence shortening and downstream sequence reconstruction. Results are computed over 100 randomly sampled query–document pairs from each benchmark. The mean block-wise Pearson correlation is 0.6525 on BRIGHT and 0.7158 on Industrial Search.

Table 10: Rank and selection fidelity of the first-order surrogate to exact counterfactual importance. Spearman measures block-wise rank agreement. Overlap@b denotes the fraction of protected blocks shared by the first-order and exact top-P selections at retention ratio b.

Dataset	**Spearman**	**Overlap@0.3**	**Overlap@0.5**	**Overlap@0.7**
BRIGHT	0.7914	0.8537	0.8764	0.8693
Industrial Search	0.8011	0.8734	0.8995	0.8933

correlation between $\{I_j\}$ and $\{I_j^*\}$ across its compressible blocks. Figure 3 reports the per-instance correlations, with mean values of 0.6525 on BRIGHT and 0.7158 on Industrial Search.

Because Protect-or-Pool ultimately depends on the ranking of block importance rather than its absolute scale, we further evaluate rank and selection agreement. The mean Spearman correlation between the first-order and exact importance scores is 0.7914 on BRIGHT and 0.8011 on Industrial Search. We additionally compare the sets of blocks selected by the two importance measures at retention ratios $b \in \{0.3, 0.5, 0.7\}$, where Overlap@b denotes the fraction of protected blocks shared by the two top-P selections. As shown in Table 10, the overlap remains between 85.37% and 87.64% on BRIGHT and between 87.34% and 89.95% on Industrial Search across the evaluated budgets.

Together, these results show that, although the same-length first-order surrogate does not exactly reproduce the magnitude of the counterfactual effect induced by sequence-shortening Protect-or-Pool, it preserves the relative ordering of block importance and yields highly consistent protection decisions. This supports its use as an efficient localization supervision signal without requiring block-wise compressed forward passes during training.

C Additional Analysis of Adaptive Budget Allocation

C.1 Controlled Efficiency of Adaptive Allocation

Average retention measures the amount of intermediate representations propagated through deeper layers, but does not by itself determine realized computational efficiency. We therefore directly compare the final AREA variants under a shared HuggingFace implementation. All models are evaluated on the same BRIGHT inputs using a single NVIDIA A800 80GB GPU with batch size

Table 11: Controlled efficiency of the final AREA variants on BRIGHT. All variants use the same HuggingFace implementation on a single NVIDIA A800 80GB GPU with batch size one. Latency is pure model-forward time per query–document pair, and speedup is measured relative to AREA (Full).

Method	Ret. ↓	nDCG@10 ↑	Lat. (ms) ↓	Speedup ↑
AREA-9B (Full)	1.00	30.10	89.30	1.00×
AREA-9B (Cumulative, $k_x = 0.8$)	0.63	29.97	73.89	1.21×
AREA-9B (Cumulative, $k_x = 0.6$)	0.49	29.33	66.98	1.33×
AREA-9B (Adaptive)	0.49	30.21	67.21	1.33×

one, and we report pure model-forward latency per query–document pair. The compressed variants include the overhead of importance prediction, budget prediction where applicable, and Protect-or-Pool execution.

AREA (Adaptive) achieves a $1.33\times$ model-forward speedup over AREA (Full) while maintaining comparable ranking effectiveness. The default Cumulative configuration ($k_x = 0.8$) obtains a $1.21\times$ speedup at an average retention ratio of 0.63.

At matched average retention of 0.49, Cumulative ($k_x = 0.6$) and Adaptive require 66.98 and 67.21 ms per query–document pair, respectively, while achieving 29.33 and 30.21 nDCG@10. Thus, Adaptive improves ranking effectiveness by 0.88 nDCG@10 under nearly identical controlled model-forward latency.

C.2 Budget Predictor Input Ablation

We further examine whether simpler instance-level representations provide sufficient information for predicting retention budgets. Specifically, we replace the block-importance distribution used by AREA with either document length or the last-token hidden state, while keeping the remaining budget learning procedure unchanged.

Figure 4 shows the evolution of the predicted budgets during training. For both alternatives, the mean budget gradually moves toward a stable operating point, while the variation across query–document pairs rapidly vanishes. The standard deviation of the predicted budgets falls below 10^{-2} and remains small thereafter, indicating that both predictors converge to nearly constant outputs rather than maintaining meaningful instance-level variation. The last-token representation exhibits a slower collapse than document length, but ultimately shows the same behavior.

Consistent with this behavior, both alternatives underperform the importance-distribution-based budget predictor in ranking effectiveness (Table 2). Under our predictor architecture and training setup, document length and the last-token hidden state therefore provide less effective inputs for learning instance-specific retention than the distribution of localization scores.

C.3 Validity of Candidate-Wise Counterfactual Sufficiency

The budget supervision in Section 3.4 evaluates one candidate at a time against a fixed-full reference, whereas inference compresses all candidates jointly. Candidate-wise sufficiency therefore does not provide a formal guarantee under simultaneous compression. We directly quantify this approximation using the seed-42 AREA (Adaptive) checkpoint reported in the main BRIGHT results.

For each query, we first disable compression and obtain the full-context score vector $\mathbf{z}^{\text{full}}$. We then run normal Adaptive inference to obtain the jointly compressed scores $\mathbf{z}^{\text{joint}} = [z_1(\hat{b}_1), \ldots, z_n(\hat{b}_n)]$. For candidate d_i, we construct

$$\mathbf{z}_i^{\text{single}} = [z_1^{\text{full}}, \ldots, z_i(\hat{b}_i), \ldots, z_n^{\text{full}}],$$

and consider its predicted budget candidate-wise sufficient when

$$R_i(\mathbf{z}_i^{\text{single}}, \mathbf{y}) \geq R_i(\mathbf{z}^{\text{full}}, \mathbf{y}) - \tau,$$

with $\tau = 0.001$. Among candidates satisfying this criterion, we define a joint violation when

$$R_i(\mathbf{z}^{\text{joint}}, \mathbf{y}) < R_i(\mathbf{z}^{\text{full}}, \mathbf{y}) - \tau.$$

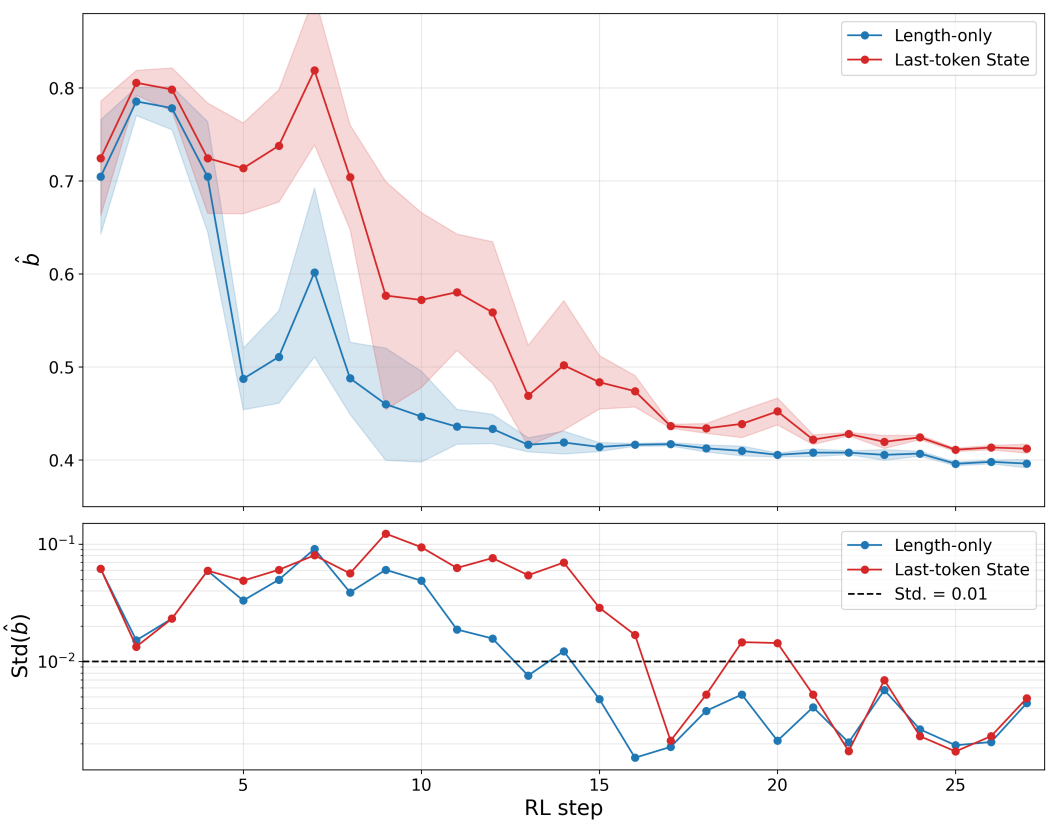


Figure 4: Budget predictions during training using document length or the last-token hidden state as the predictor input. **Top:** mean predicted retention budget across instances, with shaded regions indicating the corresponding standard deviation. **Bottom:** standard deviation of predicted budgets on a logarithmic scale. For both inputs, the variance progressively collapses and falls below 10^{-2}, resulting in nearly constant instance-level budgets.

Table 12: Validation of candidate-wise sufficiency under joint compression on BRIGHT. Full-context and joint-compression results use the same final AREA (Adaptive) checkpoint. Joint Violation is conditioned on candidates that satisfy the candidate-wise sufficiency criterion. nDCG@10 is reported in percentage points.

	Individual Suff.	Joint Violation	Mean Violation	Median Violation	P90 Violation	Full nDCG@10	Joint nDCG@10
BRIGHT	95.26%	2.38%	0.1314	0.0109	0.5100	30.49	30.21

Candidate-level diagnostics are computed on the 661 of 1,384 BRIGHT queries that contain at least one in-pool positive, corresponding to 66,100 query–document candidates. The remaining 723 queries contain no in-pool positive and therefore do not admit the listwise candidate-level diagnostic used above. Ranking-level metrics are computed over all 1,384 queries.

As shown in Table 12, 95.26% of evaluated candidates satisfy the candidate-wise criterion, and only 2.38% of these become insufficient when all candidates are compressed jointly. Equivalently, 97.62% of candidate-wise sufficient cases remain sufficient under joint execution. The joint-violation rate is also consistently low across BRIGHT domains, ranging from 0.36% to 4.03%. Although the violating cases exhibit a right-tailed reward deviation, their median violation is only 0.0109. At the ranking level, joint compression achieves 30.21 nDCG@10 compared with 30.49 under full-context execution of the same Adaptive checkpoint. These results indicate that candidate-wise counterfactual sufficiency is not an exact joint guarantee, but provides a close empirical approximation under the actual joint-compression setting.

Table 13: Training stability on BRIGHT across three RL-stage random seeds. All runs start from the same pre-RL checkpoint and use matched training data and optimization settings. The seed-42 AREA (Adaptive) run corresponds to the checkpoint reported in the main BRIGHT results. Mean and standard deviation are computed across the three runs.

Method	Seed 42	Seed 420	Seed 4200	Mean ± Std.
Fixed ($b = 0.5$)	29.21	29.25	29.22	29.23 ± 0.02
AREA (Adaptive)	30.21	30.24	30.12	30.19 ± 0.06

Table 14: Domain-stratified paired bootstrap analysis on BRIGHT using the seed-42 AREA (Adaptive) checkpoint reported in the main results. We perform 10,000 query-level bootstrap resamples over all 1,384 queries, independently within each of the 12 domains, and recompute the macro-average nDCG@10. Confidence intervals are percentile intervals for paired differences.

Comparison	Adaptive	Baseline	Δ nDCG@10	95% CI
Adaptive − Fixed ($b = 0.5$)	30.21	29.21	+1.00	$[+0.27, +1.73]$
Adaptive − Cumulative ($k_x = 0.6$)	30.21	29.33	+0.88	$[+0.18, +1.59]$

C.4 Statistical Reliability of Adaptive Budgeting

We further evaluate whether the improvement from adaptive budgeting is robust to both optimization randomness and evaluation-set sampling variation.

Training Stability. We use seed 42 as the default RL-stage random seed for the main experiments. To evaluate sensitivity to optimization randomness, we additionally repeat the RL stage with seeds $\{420, 4200\}$ while keeping the training data, pre-RL initialization, and optimization settings fixed. The seed-42 AREA (Adaptive) run is therefore the checkpoint reported in the main BRIGHT results, rather than a checkpoint selected from the multi-seed runs. Table 13 reports BRIGHT nDCG@10 for AREA (Adaptive) and fixed-ratio compression with $b = 0.5$. AREA (Adaptive) achieves 30.19 ± 0.06 nDCG@10, compared with 29.23 ± 0.02 for the fixed-budget baseline. Adaptive budgeting outperforms fixed-ratio compression in all three runs, with per-seed improvements of $+1.03$, $+0.96$, and $+0.90$ nDCG@10, respectively.

Paired Bootstrap Analysis. We additionally quantify uncertainty with respect to the BRIGHT evaluation queries using the seed-42 AREA (Adaptive) checkpoint reported in the main results. For each of 10,000 bootstrap replicates, queries are sampled with replacement independently within each of the 12 BRIGHT domains. The same sampled queries are used for both methods in each comparison, and the macro-average nDCG@10 across domains is recomputed for every replicate. The analysis uses all 1,384 BRIGHT queries, aligned across the compared methods.

As shown in Table 14, the improvement of AREA (Adaptive) over fixed-ratio compression is $+1.00$ nDCG@10, with a 95% bootstrap confidence interval of $[+0.27, +1.73]$. Compared with the approximately retention-matched Cumulative baseline ($k_x = 0.6$), Adaptive improves nDCG@10 by $+0.88$, with a 95% confidence interval of $[+0.18, +1.59]$. Both intervals exclude zero, indicating that the observed advantages are stable under query-level resampling.

Together, these analyses address two complementary sources of uncertainty: the multi-seed experiment evaluates sensitivity to training randomness, whereas the paired bootstrap evaluates sensitivity to the composition of the BRIGHT test queries. The improvement of adaptive budgeting over fixed-ratio compression is consistent across all three training runs, and its gains over both fixed and retention-matched cumulative allocation remain positive under paired query-level resampling.

C.5 Fidelity of Directional Budget Search

The directional search in Section 3.4 does not assume that ranking reward is monotonic with respect to the retention budget, and therefore is not guaranteed to recover the globally minimum sufficient configuration. We evaluate the fidelity of this procedure against exhaustive evaluation over a 0.05 retention grid on the RL training distribution, where the search is used to construct budget supervision.

Table 15: Fidelity of directional budget search on sampled RL training queries. Budget error is computed using executed retention ratios. Avg. evaluations includes the initial Budget Head proposal.

Candidates	Within 0.05 ↑	Budget MAE ↓	Non-monotonic ↓	Avg. Evaluations ↓
543	91.90%	0.0209	9.02%	7.98

We sample 100 complete query groups and retain 543 diagnostic candidates. For each candidate, all other documents in the same query remain at their full-context scores, preserving the listwise reward construction used during training. We exhaustively evaluate requested budgets $\{0.25, 0.30, \ldots, 1.00\}$, each mapped through the actual Protect-or-Pool execution rule, and define the grid reference as the minimum sufficient budget on this 0.05 grid. We compare it with the smallest confirmed sufficient budget returned by directional search.

Directional search produces an executed budget within 0.05 of the exhaustive grid reference for 91.90% of candidates, with a mean absolute difference of 0.0209. The median difference is zero and the 90th percentile is 0.0444. At the same time, directional search requires 7.98 compressed evaluations on average, compared with 16 requested operating points for exhaustive grid evaluation. We observe non-monotonic sufficiency patterns for 9.02% of candidates, confirming that budget sufficiency is not universally monotonic. These results support directional search as a lower-cost approximation to fine-grained exhaustive budget evaluation while motivating our use of the term *smallest confirmed sufficient budget* rather than a globally minimum budget.

C.6 Analysis of Instance-Specific Budget Allocation

Distribution of predicted budgets. We first examine whether adaptive budgeting produces meaningful instance-level variation rather than collapsing to a nearly constant operating point. Figure 5 shows the distribution of predicted retention budgets on BRIGHT. The predictions span a broad range of operating points, with a mean of 0.4866 and a median of 0.4915. Although the distribution is concentrated around moderate retention, a substantial number of query–document pairs receive either considerably smaller or larger budgets. This behavior differs from the length-only and last-token predictors in Appendix C.2, whose prediction variance progressively collapses during training, and provides direct evidence that AREA maintains instance-specific budget variation at inference time.

Qualitative analysis. Figure 6 provides two representative examples. In the first case, relevance evidence is distributed across multiple aspects of the document, and AREA preserves representations across these regions. In the second case, the evidence required to establish relevance is concentrated in a small number of directly answer-bearing spans, allowing substantially more surrounding content to be pooled. These examples illustrate how instance-specific retention can adapt to differences in the amount and distribution of relevance evidence.

D Dataset and Evaluation Details

D.1 Industrial Search Benchmark

We evaluate AREA on a manually annotated industrial search benchmark constructed from real-world search logs. The test set contains 300 queries and 1,728 query–document pairs. The query set includes both randomly sampled queries and medium- and long-tail queries, covering realistic search scenarios with varying intent specificity and difficulty. All query–document pairs are manually annotated by professional relevance assessors.

Relevance is annotated on a four-level ordinal scale: 3 (Excellent), 2 (Good), 1 (Fair), and 0 (Irrelevant). Excellent documents fully satisfy the query information need; Good documents satisfy most of the need with only minor omissions or deviations; Fair documents are partially relevant but miss important information; and Irrelevant documents do not satisfy the query or contain conflicting information. Table 16 reports the label distribution of the test set.

We report nDCG@10 for ranking quality and normalized Positive–Negative Ratio (nPNR) for pairwise ranking consistency. Let N_+ and N_- denote the numbers of correctly and incorrectly ordered

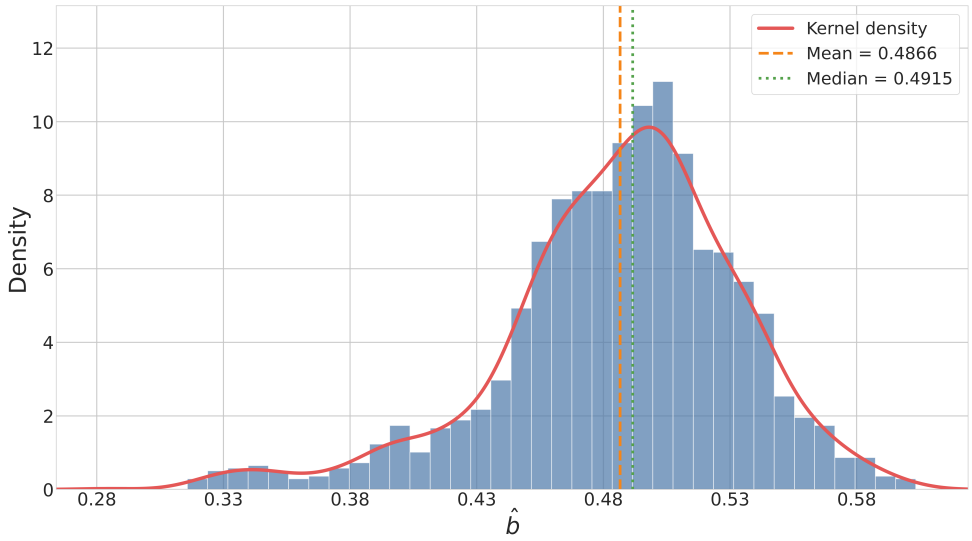


Figure 5: Distribution of instance-specific retention budgets predicted by AREA on BRIGHT. The dashed and dotted vertical lines denote the mean (0.4866) and median (0.4915) predicted retention ratios, respectively. The learned allocator produces substantial variation across query–document pairs rather than collapsing to a single operating point.

Table 16: Label distribution of the industrial search benchmark.

	Excellent (3)	**Good (2)**	**Fair (1)**	**Irrelevant (0)**
Test set	13.60%	32.18%	23.67%	30.56%

document pairs, respectively. We compute

$$\mathrm{nPNR} = \frac{\mathrm{PNR}}{1+\mathrm{PNR}} = \frac{N_+}{N_+ + N_-}.$$

Higher nPNR indicates more consistent pairwise ordering. The industrial evaluation set is held out from all training stages.

D.2 BRIGHT-Long Construction

BRIGHT-Long is constructed from the official BRIGHT queries and document collections to evaluate reranking over substantially longer candidates while preserving the original retrieval task and relevance annotations. We retain the original query text and rebuild the candidate corpus separately for each of the 12 BRIGHT domains. Depending on the structure of the underlying collection, we use one of three corpus-construction schemes.

Long-document corpus construction. For eight domains whose original BRIGHT documents correspond to chunks of longer source documents—Biology, Earth Science, Economics, Pony, Psychology, Robotics, StackOverflow, and Sustainable Living—we merge consecutive chunks from the same source document into longer blocks. Chunks are kept in their original order and concatenated until reaching a target length of approximately 1,024 Qwen3.5-9B tokens. The identifier of the first constituent chunk is used as the identifier of the resulting block.

For AoPS, LeetCode, and TheoremQA-Theorems, the original documents are self-contained items rather than chunks of a longer source document. We therefore retain sufficiently long original documents without merging them. The corresponding length thresholds are 384, 448, and 320 Qwen3.5-9B tokens, respectively. TheoremQA-Questions is constructed analogously with a threshold of 256 tokens and is retained as a short-reference subset. All 12 domains, including TheoremQA-Questions, are included in the reported BRIGHT-Long macro average.

Case 1: Distributed relevance evidence

Query. How feasible is a city with no cars and no streets? Saudi prince, Mohammand bin Salman, announced a future city spread in 170 km with no cars and no street. But any place to meet daily needs like schools, medicine and office will not take more than 5 minutes. Further, they wanted to achieve these things in sustainable way, how feasible is this idea?

Document excerpt. Another selling point of compact cities is that they are supposed to be sustainable developments. However, recent studies suggest that these developments are not as green as promised. For example, the study discusses three criticisms of the compact city model. The first criticism was that supply and demand alignments are not feasible in some cases. What this means is that there are limits to how many people can fit into a space because of resource and design limitations. Additionally, while a high density may be achievable, it likely will counter the demands and expectations of residents who want more privacy and more personal space. The second criticism of the compact city is that to produce highly desirable living conditions and a high quality of life, the cost of living needs to increase. Consequently, integrating desirable traits into compact cities makes the living spaces expensive, pricing out lower-income families and pushing them to the outskirts of town. This stimulates urban sprawl and places the burden of longer commutes on low-income workers, further expanding disparities in wealth and quality of life. The third criticism of the compact city is that its environmental impacts are significant. Dense populations mean waste and pollution are also dense. This concentrates the impact of communities and necessitates expensive control mechanisms.

Observation. Relevant evidence is distributed across multiple parts of both the query and the document, covering resource constraints, living conditions, affordability, urban sprawl, and environmental impact. AREA therefore preserves representations across multiple regions.

Case 2: Concentrated relevance evidence

Query. Can a woman give birth to twins with different fathers? Is it possible for a woman to conceive from two different men and give birth to half-siblings?

Document excerpt. Superfecundation is the fertilization of two or more ova from the same cycle by sperm from separate acts of sexual intercourse, which can lead to twin babies from two separate biological fathers. The term superfecundation is derived from fecund, meaning able to produce offspring. Homopaternal superfecundation is fertilization of two separate ova from the same father, leading to fraternal twins, while heteropaternal superfecundation is a form of atypical twinning where, genetically, the twins are half siblings – sharing the same mother, but with different fathers.

Observation. Here, both the query and the document contain substantial redundant or explanatory content, while the evidence required to establish relevance is concentrated in a small number of directly answer-bearing spans. AREA preserves these localized spans while pooling much of the surrounding content.

Figure 6: Qualitative examples of instance-specific evidence allocation. Red text denotes representations preserved at full resolution, while gray text denotes pooled content. The first example contains relevance evidence distributed across multiple parts of the query and document, whereas the second can be resolved from a small number of localized answer-bearing spans.

All document-length thresholds are measured using the Qwen3.5-9B tokenizer, which is also used by AREA. Accordingly, the target length T denotes the number of Qwen3.5-9B tokens.

Relevance Mapping. Because the document pool changes after reconstruction, the original BRIGHT relevance identifiers are mapped to the new corpus. For domains that retain original documents, gold and excluded-document identifiers are filtered by their presence in the reconstructed pool. For merged-document domains, each original chunk is mapped to the longer block containing it; a reconstructed block is considered relevant whenever it contains at least one original gold chunk. Duplicate mapped identifiers are removed.

Queries for which no relevant document remains in the reconstructed corpus are discarded. The query text itself is never modified. Relevance evaluation uses the complete mapped gold set rather than only the relevant documents retrieved within the candidate set.

Candidate Retrieval. After constructing the domain-specific document pools, we independently build a BM25 index for each domain. Following the ERank BRIGHT retrieval setting, we use Lucene/Anserini BM25 with the EnglishAnalyzer, $k_1 = 0.9$, and $b = 0.4$. The original BRIGHT

Table 17: Statistics of the BRIGHT-Long benchmark after corpus reconstruction and BM25 retrieval.

Domain	Queries	Query–Doc Pairs
AoPS	99	4,333
Biology	95	4,750
Earth Science	91	4,550
Economics	98	4,900
LeetCode	113	5,329
Pony	112	5,600
Psychology	97	4,850
Robotics	87	4,350
StackOverflow	116	5,800
Sustainable Living	101	5,050
TheoremQA-Questions	129	6,147
TheoremQA-Theorems	55	2,750
Total	**1,193**	**58,409**

query is used without query expansion. Documents listed in the mapped exclusion set are removed from the retrieved results, and up to 50 candidates are retained for reranking.

Unlike the standard BRIGHT setting, which reranks the top-100 candidates, BRIGHT-Long uses a top-50 candidate set to control the computational cost of long-document evaluation.

Dataset Statistics. The resulting benchmark contains 1,193 queries and 58,409 query–document pairs across the 12 domains, with an average of 48.96 retrieved candidates per query. Table 17 summarizes the number of retained queries and query–document pairs in each domain.

The reported BRIGHT-Long results use the macro average over all 12 domains, corresponding to the full set of 1,193 retained queries.

D.3 Public Benchmarks and Candidate Construction

BRIGHT. BRIGHT is a reasoning-intensive retrieval benchmark containing 1,384 real-world queries across 12 domains, including science, economics, psychology, mathematics, programming, and other technical domains (Su et al., 2025). Unlike conventional semantic retrieval benchmarks, relevance in BRIGHT often depends on identifying latent constraints or performing intermediate reasoning between the query and candidate document.

We use all 12 BRIGHT domains. For each query, we use the original query text without query expansion and retrieve the top-100 candidates with BM25 using Pyserini. Documents in the benchmark-provided exclusion set are removed before forming the final candidate list. All rerankers operate on the same retrieved candidate sets. We evaluate the resulting rankings using nDCG@10.

BEIR. We additionally evaluate cross-domain generalization on five datasets from BEIR (Thakur et al., 2021): Climate-FEVER, NFCorpus, SciFact, TREC-COVID, and Touché-2020. These datasets cover substantially different retrieval settings. Climate-FEVER evaluates retrieval of evidence for real-world climate-related claims; NFCorpus focuses on biomedical information needs; SciFact requires retrieving scientific evidence for scientific claims; TREC-COVID evaluates scientific-literature retrieval for COVID-19 information needs; and Touché-2020 evaluates retrieval of relevant and persuasive arguments.

For each dataset, we use its original test queries and corpus and construct a top-100 candidate set with BM25 implemented in Pyserini. AREA and all compared rerankers receive the same query–candidate pairs, and the first-stage BM25 score is not combined with the reranker score. We report nDCG@10 for each dataset and their macro average.

Retrieval Protocol. The public reranking candidate sets follow the standard BM25-based evaluation protocol used in recent LLM reranking studies: retrieval is performed once before reranking, and the reranker only changes the ordering of the retrieved documents. We use the original bench-

mark queries throughout, without reasoning-based query expansion or hybrid fusion between BM25 and reranker scores. No BRIGHT or BEIR evaluation examples are used in AREA training.